Technische Universiteit Eindhoven

Faculty of Electrical Engineering
Section of Information and Communication Systems

Master's Thesis:

Touch Screen Based Measuring Equipment
Design and Implementation

B.G.F.A.W. van Oversteegen

Coach:	Dr.ir. A.C. Verschueren
	H. van Broekhuyzen (Philips EED, sectie MMM)
	Ir. R.C.H.M. Overkamp (Philips EED, sectie MMM)
Supervisor:	Prof.ir. M.P.J. Stevens
Examiner:	ir. G.L.J.M. Janssen
Date:	10 April 1998

The Faculty of Electrical Engineering of Eindhoven University of Technology does not accept any responsibility regarding the contents of Master's Theses

Preface

Most people only read the first pages of a thesis report. Therefor this is the right place to thank all people who made my thesis a success.
During the conduction of my thesis I was fortunate to have the most able guidance of two individuals, Henk van Broekhuyzen and Ad Verschueren in carrying out their task as coach. Henk van Broekhuyzen was my coach at the Philips Equipment Engineering, Division section Measuring, Magnetizing and Matching (MMM). Ad verschueren was my coach at the Eindhoven University of Technology, faculty of Electrical Engineeing Section of Digital Information Systems (ICS/EB). Both people not only coached me on my thesis but also handed me the knowledge to make proper decisions for the future.
Furthere more I like to thank Ru^{2}d van Mullekom and Jos van Cranenbroek for their support and information about MMM equipment, Nico de Boer for killing my test PC during the first flash over tests, and Anton Vervoort for its knowledge support about ESD and EMC.
Special thanks to Ir. G. Janssen, working at the University of Eindhoven. I was very honoured that he was willing to read my paper for approval.
I acknowledge with gratefulness the help of Theo Veltman, with whom I shared a room at Philips. He showed to be a great source of information.
Finally, I am grateful to Ir. Richard Overkamp and Prof. Stevens. They offered me the opportunity to conduct this thesis containing both hard- and software components.
There is one person who I do not thank. This man is ..... right BiLLL GaTeSSSS whose software has a mind on its own.

Bertrand Oversteegen
Eindhoven, The Nederlands
April, 1998

Contents

1. Introduction *3*

1.1. The cathode ray tube **3**

1.2. MMM equipment **4**

1.3. Problem statement **5**

1.4. Objective **5**

1.5. Organisation of this report **6**

2. Man Machine Interface *7*

2.1. Introduction **7**

2.2. The User Profile **7**

2.2.1. Psychological Characteristics of user 8

2.2.2. Knowledge and experience of user 8

2.2.3. Job and Tasks 8

2.2.4. Tools for the user 9

2.3. Selecting dialog style for MMM equipment users. **9**

3. Input and Output Devices *11*

3.1. Input device **11**

3.2. Output **12**

4. Touch screen style guide *13*

Selection strategy **13**

4.1. Target Location **13**

4.2. Target Size **13**

4.3. Target Layout **14**

4.4. Touch screen position **14**

5. EMC/ESD Constraints *15*

6. Touch Screen technology *16*

6.1. Choice of Touch Screen **16**

6.2. EMC/ESD test **17**

7. Demonstrator design *18*

7.1.1. OMT 18

7.1.2. The OMT Object Model 18

7.1.3. The OMT Dynamic Model 18

7.1.4. The OMT Functional Model 19

7.1.5. Development activities 19

7.1.6. Conceptualisation 19

7.1.7. System boundary 19

7.1.8. Actors 20

7.1.9. Use-case 20

7.1.10. Context diagram 22

7.1.11. Problem statement 22

7.2. Domain analysis **23**
7.2.1. Candidate classes 23
7.2.2. Model dictionary 24
7.2.3. Associations 25
7.2.4. Attributes and Operations 26
7.2.5. Rational Rose 28
7.3. Generalisation **28**
7.4. Application analysis **30**
7.4.1. Scenario 30
7.5. System design **33**
7.5.1. Code generation 34
7.5.2. Classes 34
8. Conclusions and Recommendations ***42***
8.1. Conclusions **42**
8.2. Recommendations **43**
references *44*

Appendix

A.	**User profile Checklist**	**51**
B.	**Job and Task Characteristics**	**52**
C.	**Dialog Styles**	**53**
D.	**Evaluation form**	**66**
E.	**Touch Screen technologies**	**67**
F.	**Use-Case**	**77**
G.	**List of candidate classes**	**81**
H.	**Model dictionary**	**82**
I.	**Questionnaire**	**83**
J.	**Header file of tube class**	**84**
K.	**C code file of tube class**	**90**
L.	**Touch screen producers**	**91**
M.	**OMT notations**	**92**
N.	**Design**	**93**
O.	**internal scenarios**	**94**
P.	**Dialogs**	**99**
Q.	**Example of part from parameter file**	**102**

Abstract

Today a lot of user interfaces are redesigned to satisfy the needs of industrial operators. Man machine interfaces are optimised to offer a great functionality to the users. The main subject of this thesis was to investigate the possibility to redesign an existing user interface for industrial production equipment. This comprises three major parts.

The first part deals with the selection of appropriate dialogs to use when operating with industrial equipment. A comparison is made between all possibilities offered by several dialog styles. The results show that a menu system with direct manipulation is highly recommended, to satisfy the needs for a big group of different users. The literature also shows that de desired dialog is best performed with the utilisation of a touch screen input/output device.

The second part handles about the constraints of user interfaces, if a touch screen is used as a communication media. A comprehensive inquiry has been made to design guidelines for a touch screen biased user interface.
The electrical characteristics of a touch screen device are important. Industrial environment is very harsh for delicate electronics. For present study the electric static discharges during measuring were of great importance. The surface acoustic wave technology proved to be reliable touch screen technology in those harsh environments.

Finally the last part handles about the design of a touch screen based industrial equipment. An Object Oriented method is used to design the measuring system. The implementation is done with Visual C++. The design is made flexible and reusable to satisfy the demands of future equipment. The dialogs used are designed using the style guide made in the second part of the thesis.

1. Introduction

Main subject of this thesis: develop a touch screen user interface for control of colour CRT production equipment. In this introduction chapter we will describe the CRT manufacturing process, pose a problem statement, specify the objectives and explain the organisation of this report.

1.1. The cathode ray tube

The colour cathode ray tube (CRT) found in television, computer and video monitors utilise a shadow mask or aperture grill a fraction of an inch (1/2" typical) behind the phosphor screen to direct the electron beams for the red , green, and blue video signals to the proper phosphor dots. Since the electron beams for the R, G, and B phosphors originate from slightly different positions (Individual electron guns for each) and thus arrive at slightly different angles, only the proper phosphors are excited when the purity is properly adjusted and the necessary magnetic field free region is maintained inside the CRT. Note that purity determines that the correct video signal excites the proper colour while convergence determines the geometric alignment of the
3 colours. Both are affected by magnetic fields. Bad purity results in incorrect colours. Bad convergence results in colour fringing at edges of characters or graphics.

The shadow mask consists of a thin steel or InVar (a ferrous alloy) sheet with a fine array of holes, one for each trio of phosphor dots - positioned about 1/2 inch behind the surface of the phosphor screen. With some CRTs, the phosphors are arranged in triangular formations called triads with each of the colour dots at the apex of the triangle. With many television and some monitors, they are arranged as vertical slots with the phosphors for the 3 colours next to one another.

The location at which the electrons hit the phosphor (landing) and the relative position of the three separate electron bundles (convergence) is determined by the mechanical parts of the electron gun.

The following is a greatly simplified description of the general process of colour (shadow or slot mask) CRT construction.

- The screen and envelope glass pieces are molded separately and then glued together as one of the last steps of assembly prior the baking and evacuation.

- The shadow mask is manufactured through a photo etching process. Since a position error of even a tiny fraction of a mm would result in purity errors, each shadow mask is unique for its faceplate. They are not interchangeable.

- The CRT is evacuated. This takes several hours at the vacuum pumps. The assembly is then sealed by heating and melting.

- The getter - part of the electron gun assembly - is then 'activated' via induction heating from a coil external to the neck of the CRT. This vaporises and deposits a highly active

metal on the interior of the glass of the neck. The getter material adsorbs much of any remaining gas molecules left over from the evacuation of the CRT.

- When the CRT is ready it is measured to investigate if there are any loose contacts or if there are conducting paths that should not exist.

- The next step is to calibrate the three separate electron beams (red green blue) on the appropriate phosphors dots. The CRTs of Philips are equipped with an IMACO ring at the front of the electron gun. The IMACO ring is basically a programmable 8-pole magnet. The programming is done with the aid of 8 magnetic coils which magnetise the IMACO ring.

- Finally the CRT is matched with a deflection coil that provides optimum purity. It takes some ingenuity to get a good match between the light used for exposure and the future electron optical system, in order to get good purity.

The mentioned technology used in CRTs has not dramatically changed during its existence.

1.2. MMM equipment

The Measuring, Magnetising and Matching (MMM) is done with equipment, developed and build at the MMM section of Philips. The section MMM is part of the Equipment Engineering Division (EED) of Philips. The EED develops and builds equipment to produce CRTs. CRTs are produced in a lot of countries all over the world. Every country has its own characteristics about wages, education, culture etc. Because of the amount of produced CRTs it is important that the costs are as low as possible. This means that in countries where labour is cheap most of the activities have to be done by people. On the other hand if labour is expensive a fully automated production line is the most profitable one.

MMM offers a scale of equipment to perform the desired measurements. From hand operated equipment to fully automated production lines. Not only equipment for the three disciplines is made but also equipment to perform quality measurements on finished CRTs.

Most of the equipment projects images on the screen for evaluation. The more automated equipment has vision capabilities to interpret the projected picture. The low automation equipment leaves the interpretation to the operator. This certainly does not imply that a CRT calibrated with a fully automated equipment performs better than a CRT calibrated manually. The calibration always is a mean over the entire screen. The middle of the screen and the corners are each opposite. The electron beam has to travel a log way to hit the phosphor in the corner. Making the perfect image in the corners generally gives a miserable image in the middle of the screen.

The equipment that projects images on the screen has high tension power supplies to activate the electron beams. Not only the high tension voltages put constraints on equipment design but also the fast switching modes of those power supplies. Most of the CRTs are tested under stress (higher voltage than during normal operation). The consequence is that flash-overs are normal (although not desired) during the measuring of the CRTs.

Because of the fact that there has not been any drastic technology change in CRTs the equipment to calibrate them hasn't changed very much either. Most of the newly designed equipment were upgrades on older ones. Not only the lack of technology change is responsible, but also the existing equipment is very stable and accurate.

1.3. Problem statement

The variety in MMM measuring equipment is very high. There is fully automated test- and calibration equipment but also equipment that is operated manually. Due to the upgrade policy a great variety in input output devices are used to operate the equipment. Varying from VT220 terminals to push button panels and from VGA monitors to Led displays. A touch screen could replace this mixture.

The architecture for the most complex equipment is based on one or two VME systems, together with 68K processors. Sometimes the mechanical movements are controlled by a PLC. The intention is to migrate the measuring systems to PC based platforms.

Electrostatic charge of operators and electromagnetic radiation from the equipment may damage the new equipment. This is one of the greatest concerns of the section MMM. Currently if a flash-over is detected the system is rebooted to avoid errors in the proceedings of the calibration. Booting may only consume a little time because the amount of CRTs is high so stalling the production costs money.

A PC based platform requires new development environments. The section MMM has chosen to migrate to Windows NT. This means a Microsoft environment for future developments.

1.4. Objective

The objective of this study is to obtain more information about novel approaches to user interfaces and system design. This is part of a broader study to investigate the feasibility for the section MMM to design a new generation of MMM equipment. Important aspects in this study are:

- Desired functionality for the users.
- Description of man machine interfaces (MMI) for middle to large sized MMM equipment.
- Information model of MMI
- Market scan of available touch screens
- Selection of suited touch screen technology
- Style guide for MMI with touch screen
- Verification of ESD and EMC requirements
- Designing an implementation of a working application in a windows environment.
- Evaluation of the application.

1.5. Organisation of this report

The organisation of this report is as follows. Chapter 2 handles about the Man Machine Interface (MMI) applied in MMM equipment. A thorough investigation is added in the appendix C about the possibilities in MMI's. This data is mapped on the characteristics of the operators using the MMM equipment. This strategy points out which MMI is applicable in the user's point of view. Based on the results, chapter 3 handles about the selection of the IO devices appropriate to optimise performance of the chosen MMI. As the title of this report proves a touch screen is a well-suited IO device to support the optimal MMI. The use of a touch screen in a graphical user interface puts certain constraints on the design. These constraints are pointed out in chapter 4. As mentioned in the problem statement ESD and EMC are of great concern when introducing new parts of the equipment. In chapter 5 the ESD and EMC constraints are evaluated to make a good decision what touch screen technology is appropriated for MMM equipment. Chapter 6 evaluates all possible touch screen technologies. Together with the desired constraints a touch screen technology is chosen to be used for MMM equipment. Chapter 7 shows the design of a demonstration model of a MMM equipment using a touch screen. The demonstrator is designed to serve as a blue print for other MMM equipment. Rational Rose (OMT) is used as case tool to design the demonstrator. Finally chapter 8 contains the conclusions of the study.

2. Man Machine Interface

2.1. Introduction

Industrial systems are really more than the software alone and the scope and purpose of the system is wider than the functionality provided by the software. In fact, the software is only one component in a larger system and provides only a subset of the functionality that is desired. This larger system includes at least one human user and may, in fact, include other users and other software systems.
Considering the two main subsystems, the computer and the human, a few initial observations can be made. The human is flexible and adaptable. Most importantly, the human can learn how to operate in new environments. By contrast, the computer system is not flexible or adaptable; at least most commercial systems currently are not. Inputs must be made in particular format, outputs are predefined. Thus, for a given computer system the human can learn and adapt while the computer cannot. On the other hand, people build computers, and computers can be redesigned.
In the past, the design of the MMI was heavily biased to accommodate the weaknesses of the computer system. More recently, due to advances in the technology, the bias toward accommodating the deficiencies of computers in the MMI is slowly shifting. Initially, the idea has been to shift the responsibility for the success of the MMI interaction to the computer designer. Or better yet to the MMI specialist. That is, without actually making computers more adaptable or more flexible, it is nevertheless possible to make them more compatible with the way people work and communicate.
The touch screen feasibility study is part of a research project to use new technology so few restrictions are made on which MMI to use. The method used in a book about user interfaces [Mayhew, 1987] is adopted to guide the choice for a possible MMI.

2.2. The User Profile

Perhaps the fundamental principle, from which all others derive, is to *know the user.* It is wrong to assume that all users are alike, and that all users are like the developer. So to derive an efficient and effective MMI it is important to make an inventory of the users. User performance may be described in terms of a number of general determinants, including

- the human information processing system,
- the user's psychological characteristics,
- the user's knowledge and experience,
- the user's job and tasks,
- the user's physical characteristics,
- the user's physical environment, and
- the user's tools.

The first determinant tends to be relatively constant across individuals, while the last six may vary significantly across user groups or individuals. From the software designer's point of view, the first six are predetermined, while the last is under the designer's control.

Knowledge of human information processing strengths and weaknesses leads to a number of design goals for interactive systems. It is also true that knowledge of the characteristics of human information processing allows people to make predictions regarding human performance on interactive systems. The idea behind a model such as the Model Human Processor [Card, 1981] is that a good, validated model or theory allows us to make predictions about behaviour without actually having to collect empirical data. Unfortunately the state of science of computer-human interaction is not such that we can yet apply models such as the Model Human Processor or GOMS [Moran, 1983] with complete confidence, eliminating all need for testing. The determinants involving the human performance are summarised in appendix A and B.
The main part from the tables in the appendix are drawn from Mayhew but they are adapted to the users of MMM equipment. Most determinants are trivial but some need extra attention.

2.2.1. Psychological Characteristics of user

Designers can design interfaces to address differing cognitive styles, negative attitudes, and low motivation, or to exploit and maintain positive attitudes and high motivation, for instance, consider the following goals:

User characteristic	**Design goal**
Low motivation, discretionary use	Ease of learning
Low motivation, mandatory use	Control, power
High motivation, due to fear	Ease of learning, robustness, control, power
High motivation, due to interest	Power, ease of use

E.g. users who are highly motivated out of fear (for example, of losing their job or of appearing incompetent) need the reassurance that the system is not overly complex and will not be overly complex to learn.

2.2.2. Knowledge and experience of user

User experience in not just a simple binary dimension: novice and expert. The dimension of knowledge and experience is a continuum. A number of characteristics that are relatively independent are listed in appendix A and B. For more detailed information the reader should read the book Mathew wrote.

2.2.3. Job and Tasks

Job and task characteristics drive MMI design in many ways. Menus (system controlled-forced choice), question and answer dialogs (system-free choice) are highly structured interface styles and should only be used in tasks that are themselves highly structured (input and output modes). Command languages (user controlled-free choice) are highly flexible, unstructured dialog styles and so are appropriate for unstructured tasks (processing mode). Ease of learning should be compatible with the turnover rate and primary training. In general, the ease of use-ease of learning trade-off should be guided by frequency of use, task importance, and system usage, as in Table 1.

Frequency of use	
High:	Ease of use
Low:	Ease of learning and remembering
Task importance	
High:	Ease of use
Low:	Ease of learning and remembering
System use	
Mandatory:	Ease of learning and remembering
Discretionary:	Ease of use

Table 1 Compatibility

2.2.4. Tools for the user

The performance is affected by the kinds of tools available to the user, e.g. a user with a calculator is more effective than one who must do calculations by hand. More importantly operator performance may differ accordingly to the effectiveness of the user interfaces of the different systems.

Now the simple checklist provided in appendix A and B can be used to develop a questionnaire, which can be answered by a representative sample of users or people who know the users. The questionnaire is printed in appendix I. The questionnaire is answered by 4 people from EED who's task it is to keep in touch with the end users. The conclusions drawn from these questionnaires are displayed in table 2.
To decide what design rules are to be used and what the constraints are, first a more detailed explanation about dialog styles is appropriate. This explanation can be found in appendix C.

2.3. Selecting dialog style for MMM equipment users.

To make a selection between the dialog styles, all the analyses in the previous sections can be gathered in selection tables, see appendix D. Each cell in the table holds a particular value of the user characteristic. Using the results from the questionnaire a first scan marks several dialog styles. The next step is to look if an unmarked cell has a serious disadvantage. If not, the cell is marked. Operators for MMM equipment can be separated in two groups. The first group comprises of people working at equipment operated by the operator e.g. COPER T1. In those cases the operator is constantly interacting with the MMI of the equipment. The second group has a more controlling part. Those operators supervise one or more automatic equipment. Only if a failure has occurred or a preventive action has to be taken, the operator will interact with the MMI of the equipment. In appendix D it can be seen that for both groups the menu style and the direct manipulation are a good choice for dialog style. For the process-maintenance engineer all dialog styles are appropriate. Because of the huge rate of input and output variables used by middle to large sized MMM equipment, fill- in forms and command

language would be appropriate. A mix of Menu, Fill-in forms, Command language and Direct manipulation seems to be optimum for MMM equipment. When using a mix it is very important to apply the right style for the right dialog. Menu style and direct manipulation are most efficient for navigation, while Fill-in forms and Command language are most efficient when entering big amounts of data. Looking at the operator tables in appendix D the Fill-in forms and command language has positions with no marks. Both dialog styles should not appear in the MMI when an operator is at the controls. For the process-maintenance engineer no limitations hold.

	Operator Manual	Operator Auto.	Maint. Process
User Psychology			
Attitude	Positive	Positive	Positive
Motivation	Moderate	Low Moderate	moderate
Knowledge and experience			
Typing skill	Low	Moderate High	Moderate High
System experience	Moderate	High	High
Task experience	Moderate	Moderate High	Moderate
Application experience	High	Moderate	Moderate
Use of other systems	little or non	High	High
Computer literacy	Low	High	High
Job and task characteristics			
Frequency of use	High	Moderate	Moderate
Primary training	Little or none	High	high
System use	Discretionary	Mandatory	Highy
Turnover rate	Moderate	Low	Low
Other systems			
Task importance	Moderate	High	High
Task structures	High	Low	Low

Table 2 : User characteristics

3. Input and Output Devices

At the current time various input and output devices are available to designate data, locations, movements and signalling, including keyboards, mice, light pens, trackballs, joysticks, screen-buttons, lights, gages and touch screen devices. Voice systems are just beginning to gain popularity as input device, but for industrial environments voice systems are not applied.

3.1. Input device

In the previous section four dialog styles are chosen to be appropriate for MMM equipment. In table 3 the suitability of input devices is given for the desired dialog style [Helander, 1988; Debora, 1988].

	Menu	Direct	Fill-in	Command
keyboard	o	--	+	+
mouse	+	+	-	-
light pen	+	o	-	-
trackball	+	o	-	-
joystick	o	-	-	-
touch screen	+	+	o	-*

Table 3, Input devices for MMM equipment appropriate dialog styles

* for limited alphanumeric data entry a touch screen may prove to be useful [Plaisant, 1992]

There are three possible input devices, that might be appropriate for MMM equipment:

- keyboard
- mouse
- touch screen

A mouse in an industrial environment is by far not an optimum solution. Keyboards tend to get stuck in polluted areas. Another mayor drawback, as can be seen in the section ESD/EMC test, is the fact that keyboards are a prey to H fields present at ESD. In general, keyboards are, if possible, avoided in industrial equipment.

Touch screens do not cover all the appropriate dialog styles but, as can be seen in the Plaisant study, for limited alphanumeric data entry, a touch screen even has some advantages over a keyboard. The mayor drawback of touch screens is the target size. See chapter 4 for more information. For large data entries a keyboard is preferred to get the highest performance from the process- or maintenance engineer. If a specific MMM apparatus needs large data entries, an additional plug-in keyboard or a shielded drawer keyboard is most appropriate.

Process continuation buttons are used to navigate back and forward in (semi) manual controlled apparatus. These buttons are used intensively. For those buttons it is very important that they can be located and accessed easily. To get a reference for location they might be situated in the corners of a touch screen, but this is not an optimal solution. To get the highest performance for that control external buttons are most appropriate although the drawback with touch screen might be solved if the sensitive area around the button is relatively large.

3.2. Output

The traditional output device for MMM equipment is signalling lights, gages (analog digital), LED displays and screens. More recently, LCD's are used as an alternative output device. Every MMI is specially hardware made for a specific apparatus. When using touch screen technology the only hardware necessary is a touch screen. All output goes via the touch screen.

The main problem using a touch screen is the amount of real estate on the screen. In practice this does not seem to be a problem because of the flexibility the software panel offers (Most of the controls are made in software). During normal process operation the touch screen only has to display relevant parts during that process, so real estate is shared during the life cycle of a process. In this way apparatus space is gained because large switching panels are superfluous.

Small MMM equipment could also profit from integrating touch screen technology. In those apparatus a small LCD display with touch screen may be applied. This also holds for distributed controls in a big apparatus. In this way the maintenance engineer can control local parts of the apparatus. This is a space and costs saving solution.

4. Touch screen style guide

Accuracy of input using touch screen devices is affected by a number of variables that include device type, target size, target location, touch screen position, and selection strategy. Several studies have investigated these variables and produced sometimes complementary and sometimes conflicting results. The conflicting results, however, may in many cases be a problem of interpretation or a difference in hardware configuration. The problem of interpretation would merely disappear if a standard of interpretation could be adopted. When interpreting results from previous studies it is important to look at the touch screen technology used (resolution/parallax).

Selection strategy

Basically there are three used strategies [Potter & Weldon, 1988] which allow a user to select one of a set of displayed predefined areas. The simplest strategy, land-on, uses the initial touching of the touch screen for selection [Sears & Shneiderman, 1991]. If a selectable item is under the initial touch it is selected, otherwise nothing is selected. All further contact with the touch screen is ignored until the finger is removed. The second strategy, first-contact, was designed to work basically the same as land-on but take advantage of the continuous stream of touch data provided by the touch screen controller. Users make selection by dragging their fingers to the desired item. In this strategy it is not what position the user lands on that becomes selected, rather is is whatever selectable item the user first contacts. If the user makes first contact with some undesired item before reaching the desired item, the undesired item will be selected. The third strategy, take-off, was designed to utilise the contours stream of touch data and give more user feedback. Whenever users make contact with the touch screen, a cursor appears slightly above their finger so that the specific position of selection is known. As long as users keep their fingers in contact with the screen, no selection will be made. After dragging the cursor, when the users are satisfied with its placement, they confirm the selection by removing their finger from the touch screen.

4.1. Target Location

It is recommended that for applications having established key input areas, positions along the lower and right-hand borders of the touch screen should be used to minimise activation time and error [Bering, 1989]. Use of the lower border exclusively can accommodate users with either a right or left-hand preference. Another study only investigating accuracy and target locations revealed the highest accuracy in the top middle section (the screen was divided into 9 sections) of the screen. This was not surprisingly because using a slight (15 degrees) backward tilt of the screen, the line of sight, normal to the screen, intersected just below the centre of the top row. Parallax shifted the lower rows downward, increasing the error distance. A side result from a study concerning screen angle versus accuracy showed larger Mean Y errors at the top of the screen [Beringer, 1985].

4.2. Target Size

Only a few studies handle about target size but information about optimal target size is reported as side results in several other studies concerning touch screens. The main problem is that no study used more than 5 target sizes to retrieve their information so there is not an explicit function concerning target size verses accuracy trade-off.

One study [Leahy, 1990] developed a graph to show expected accuracy as contour lines with touch sensitive region dimensions as axes. Tree visual target sizes were used: 7.5 mm2, 12.2 mm2, and 20 mm2. As a worst case reading, they stated that a target size of 36 mm^2 would result in an accuracy between 50- and 99%. Most studies indicate that keys may be closely spaced provided the individual keys are adequately sized [Beaton *et al*, 1995]. They also state that the corresponding touch sensitive area extending beyond the visible target was found to be optimally 2.7 mm above and below target and 1.7 mm to the right and left of the target. Others [Leahy, 1990; Beringer, 1985] noticed a general bias to touch low and to the right of targets was found. If clear blunder errors could be eliminated, a target size of 18 mm in x and 15mm in y would be appropriate [Bering, 1989]. For small target selection purposes it is even possible to use 5 mm in x and 3 mm in y [Sears, 1991], but one has to use an advanced selection technique taking longer to select a target. Looking at all studies a target size of 20 x 20 mm is a good trade off between real-estate and performance.

4.3. Target Layout

Two studies investigate the design of the targets. The [Valk, 1985] study revealed that users of touch screen buttons consequently aimed at the labels or pictures on the button. The labels and pictures should be positioned away from the edge of the touchable portion of the buttons to reduce invalid touches. The same study showed that it is important to have a feedback when a touch is detected. This fastens the learning curve that is always detected in studies with touch screens. The [Egido *et al.*, 1988] study shows that the performance of the user is higher when he can make a decision based on labels and pictures. This means that equipping the button with pictures and labels will optimise the performance.

4.4. Touch screen position

Most of the studies involving touch screens mention the parallax problem. The [D. Beringer, 1989] study investigates the effect of placing the touch screen to different line of sights with the user. The results show that even a 17 degree angle altered the touch positions with 5 mm. Not only the touch location is affected by poor touch screen placement. But also the muscular fatigue is huge if the arm and hand of the user is unable to rest in-between touches. It is even optimal if the arm remains in contact with an anchoring point.

5. EMC/ESD Constraints

Electromagnetic compatibility (EMC) is the capability of an electrical device or system to operate in its electromagnetic environment without disturbing or being disturbed by it. EMC is an important criterion of product quality.

According to the definition, EMC is subdivided into electromagnetic interference (EMI) and electromagnetic immunity or susceptibility (EMS).

The EMI depends on the device that is causing the electromagnetic interference. If one part of an equipment is measured and meets the EMI requirements it will not violate those requirements if that part is used in a complete set-up.
The EMS depends on the environment. Other devices may transmit electromagnetic waves but it is also possible that interference is caused by a process in the working environment. The MMM equipment is situated on the production floor and the process involves high voltage switching and demagnetising sequences. Sometimes a jump over occurs and an electromagnetic pulse is generated, these pulses are called System Generated ElectroMagnetic Pulse (SGEMP). This is a non-repetitive interference, and in this category one can also place the Electro Static Discharge (ESD). ESD events are caused by familiar triboelectric processes [Greason, 1994], in which personnel accumulate static charge, by friction between their clothing and other objects in the workspace. When a charged person touches a metal object such as a piece of MMM equipment, the charge flows off in a spark. There are several other ways by which a person may by charged e.g. by switching on and off CRT's [Franey, 1995].
To show their conformity to the EMC requirements prescribed by law, all electrical devices have to be marked with the CE conformity mark. A manufacturer/supplier has several routes to obtain the CE marking. The first one is to place the mark on their responsibility. The manufacturer/supplier draw a (standard) certificate in which they state that the product satisfies all the European standards. The manufacturer/supplier is advised to argument the certificate in a product file. The second way is for manufacturers/suppliers who are forced to engage with a competent body.
This may happen because they want to differ from certain aspects in the standard. The third route is to test single products separately and approve them with the CE mark. Finally there is a route for manufacturers with the EN29001 - Certificate. As a consequence of the ISO9001 standard they are obliged to handle according to the standards, consequently they are only checked on the application of the Quality Assurance System.

There are a few measures that can be taken to reduce the EMI [Franey, 1995; Laan, 1993]. The first one is good PCB design. This is a measure we have to believe from the manufacturer. The second is to avoid loops or large wiring. Another important measure is the path of the wiring. If possible the wiring must follow the grounded enclosure to avoid coupling with the interfering field. It is not sufficient to use coax cables because noise currents are induced on the cable shield by the magnetic fields [Schelkunoff, 1934]. Probably the most important measurement against ESD is the grounding system. There must be no loose ends and the grounding wiring must be of excellent quality. Finally electronic or mechanical protection can be applied to the sensitive parts of the electronics. This is not always the solution because if the protection is starting to work the properties of the interfering signal are changed and the destructive properties of the signal may be even bigger. To make a good decision on the touch screen to use, it is important not only to look at the CE marker but also on the construction and technology of the touch screen.

6. Touch Screen technology

In order to select the touch technology that best fits the MMM equipment, it is important to take a brief look at how each technology functions. There are seven basic types of touch technology: capacitive overlay, force vector, guided acoustic wave, resistive overlay, scanning infrared, strain gage, and surface acoustic wave. Each type of touch technology has attributes that are desirable for specific applications.
All types of touch systems are attached to a display unit, whether a terminal, CRT, flat panel display, static graphic, or combination of flat panel display and static graphic. The difference between the technologies lies largely in the way the touch is detected and the method used to process the touch data.

Appendix E gives an introduction to the types of touch technologies available today. It is advised to read this appendix because it contains valuable information.

6.1. Choice of Touch Screen

By investigation of the touch screen systems the choice is narrowed to three technologies: surface acoustic wave, capacitive and five-wire-resistive. In appendix L there is a list of the touch screen manufacturers from which information is received for this feasibility investigation. The surface acoustic wave is exclusively manufactured by ELO. The capacitive system is manufactured by Micro Touch and the five-wire-resistive is manufactured by ELO and Micro Touch. These two manufacturers are the world leaders in touch systems. Both developed their own technologies to improve accuracy and lifetime. It is therefore that surface acoustic wave and capactive overlay are well suited in industrial environments.
According to ELO and Micro Touch all three of the touch screen technologies are unaffected by EMI from other nearby touch screens, CRTs, other displays and environmental EMI. Electrostatic Protection is covered by the IEC 801-2 standard. For the five-wire-resistive system proper transient protection must be applied to withstand high discharges. Even for the capacitive overlay an additional metallic layer is placed between the CRT surface and the active metallic layer. This additional layer protects the touch screen technology against the high ESD coming from the CRT but also contributes to the low transparency.
Looking at the technologies in combination with the EMC/ESD constraints the surface acoustic is in favour. Both 5-wire resistive and the capacitive touch systems use an overlay with a metallic coating, so there is a big surface and the Zt (surface transfer impedance) [3] is also rather big. The noise voltage induced by the magnetic field is accordingly high. In the surface acoustic wave touch screen technology the only place where noise voltage can be induced is in the transmitters, receivers and the wiring to the PCB. If the Transmitters and the PCB are well grounded the influences of an interference signal is smaller in surface acoustic wave than in both five-wire-resistive and capacitive touch screen technologies.

Another important selection criteria is the price.
A touch screen with surface acoustic wave costs:

2182 DM a 1,12 = fl 2.443,84.

The same monitor equipped with a capacitive touch screen system costs

fl 3.972,00.

So not only in technology the surface acoustic is preferred but also in price competition the surface acoustic is the preferred one.

6.2. EMC/ESD test

The selected touch screen is tested in a simulated environment. After adding one extra protection to the communication cable the touch screen did not fail if flash over tests were conducted. The complete test report is available at Philips EED section MMM.

7. Demonstrator design

Using a set of defined techniques and sign conventions to convert wishes to a software product, is called software engineering (S.E.). A S.E.-method always uses sequential predefined steps. The lifecycle of the software starts with the initial formulation of the customer's wish and continues via analysis, design, and implementation to testing. Followed by the installation and maintenance of the application.

There are a lot of tools to guide a developer through the bumpy road of software development. At the section MMM a pilot project was developed as a windows application. In the future MMM wants to continue developing in windows so for the demonstrator the window environment is used. The compiler to use is C++ which allows object oriented design. There are several software engineering methods for developing OO structured designs. The pilot project used OMT developed by Jim Rumbaugh [Rumbaugh 1991]. Another famous OO tool developer is Grady Booch [Lockheed, 1997]. Currently they are working at the Rational Company developing a method using all advantages of several development tools. The demonstrator is developed with OMT from Rumbaugh and the tool used is Rational Rose.

The equipment selected to be used for the demonstrator is the COPER T1. COPER T1 stands for COmbined PERformance Tester with 1 camera. This name indicates that different measurements can be executed. In the basic software version only the convergence measure algorithm is implemented. Options like landing, sharpness and raster measuring algorithms can be delivered as add-on measurements on the basic version. The measurements can be done on CMT and TVT colour picture CRTs with high accuracy.

7.1.1. OMT

The OMT methodology has three fundamental models, each representing a different view of the system under consideration. The three views (the object, dynamic, and functional models) allow developers to emphasise distinct aspects of the system as the circumstances direct. Associated with each model is a micro process that defines the steps necessary to develop the model.

7.1.2. The OMT Object Model

The OMT object model is the fundamental model on which the remaining models are based. The object model captures the entities that appear in the application and solution domain, their structure, and the relationships among them. Class diagrams, showing the existing classes and their relationships, along with an occasional object diagram, showing the individual instances of classes, compose the object model.

7.1.3. The OMT Dynamic Model

The OMT dynamic model indicates the dynamics of the objects and their changes in state. By exploring the behaviour of the objects over time and the flow of control and events among the objects, the dynamic model captures the essential behaviour of the system. Scenarios are

captured in a message trace diagram. These diagrams, which show the life history of objects, compose the OMT dynamic model.

7.1.4. The OMT Functional Model

The OMT functional model captures algorithm requirements, design, and data flow. The functional model is best captured in textual form as a functional specification. In the early phases of system development, use-cases capture the system's functional requirements. During later development phases, object message diagrams are used to capture design-level detail of the functional model.

7.1.5. Development activities

The OMT process is best described as a set of stages or activities that need to be performed to construct a system. Each activity involves using multiple models and has several deliverables. Figure 1 shows the high level approach to the activities.

7.1.6. Conceptualisation

To capture the system demands OMT uses use-case modelling. An use-case describes an actor's typical use of the system. An actor is a role played by a physical person or object when interfacing with the system. An actor may engage in more than one typical use case if the uses are separated in time or place.

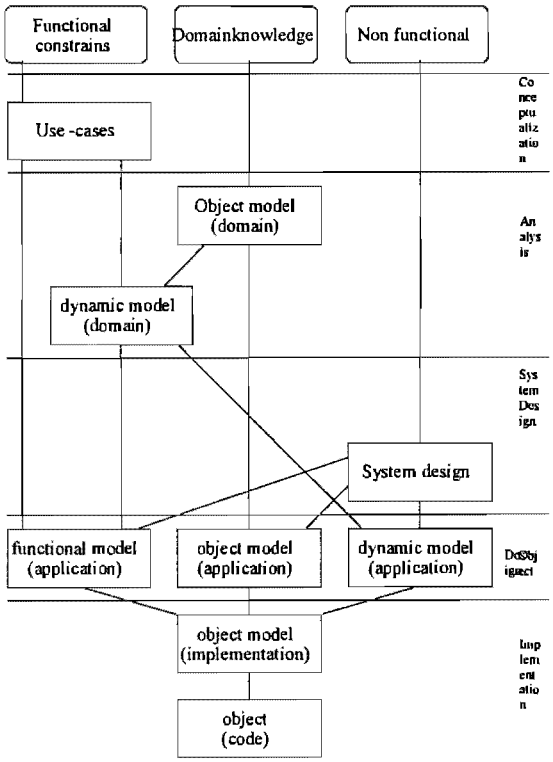


Figure 1 : OMT structure

7.1.7. System boundary

The system boundaries are defined by the responsibilities of the system. For the Coper the responsibility is to executing measurements on a CRT (TVT or CMT). The CRT can be positioned on the inside or outside of the measuring system. When generalising the system the question rises "is it a CRT that is to be measured"? By placing the CRT (measured object) outside the system borders it is easier to adapt the system for future use.

The measurements are properties of the images on the CRT. As mentioned earlier these images are taken at predefined places on the screen. A Coper can have several automated forms. It may be equipped with 12 cameras for a fully automated system, or it may be equipped with one camera. This camera may be positioned manually, or it is integrated in a semi-automatic positioning system. In the last case an XY-frame is used to position the

camera in front of the desired place on the CRT. For the measuring system it is not important how an XY-frame works. For this reason the XY-frame is placed outside the system boundary.

The complexity of MMM equipment makes it almost impossible to completely redesign a MMM equipment, and give full attention to all parts of the design. The main part of this study lies in the user interface, and how to adapt the current software to be used with the new user interfaces. The system is designed with full functionality, but at the place where specific measuring code should to be implemented a call to an existing COPER T1 is executed to retrieve the desired measurements. For simplicity the existing COPER T1 is placed outside the system boundary.

The user is also outside the system boundary. In section 2.3 one can see that there are two types of users with a different privilege in the system. One is the operator and the second is the maintenance engineer.

7.1.8. Actors

For general MMM equipment the external actors *User* and *Tube* are identified. The demonstrator needs a remote control. That remote control can be seen as an actor. For some MMM equipment a XY-Frame is needed, so that forms a fourth actor.

7.1.9. Use-case

The actor identified as User, is the one that initiates all activities in the system. There are two types of users possible. The first is the operator mostly initiating the main measurements, secondly the maintenance user. This user is able to fully use the possibilities of the measuring equipment and to freely change process and program parameters. To make this possible the user must login (1), this is the first use-case.

First the use-case of the operator is given.

- The user can select a tube file name according to the tube to be measured (2).
- The user chooses a measurement to conduct (3).
- The user chooses the field to be measured (4).
- For some measurements it is necessary to select a colour (5).
- The user must have the ability to execute a video search to place the image in front of the camera (6).
- The user must be able to execute a colour recognition to enable colour detection by the system (7).
- The user is able to execute basic measurements to detect possible errors in the system (8).
- The user is able to access the basic functions of the frame grabber to detect errors or to remedy possible errors (9).
- The user must be able to change the image type (10).
- The user must be able to select the data collection system (11).

- The maintenance user is in addition to the previous tasks able to add new users to the system (12).

- The maintenance user is able to edit the following parameters and variables: program table, program parameters, tube parameters, position parameters, colour recognition parameters, convergence parameters, eccentricity parameters and linewidth parameters. (13)
- Both users must be able to see the measurements (14).

The remote control is used to access COPER T1. The use-case for that actor can be substituted by a functional diagram with the functions available by the COPER T1 remote control access. The table containing those functions is available in the users guide of the COPER T1 [Mullekom 1996].
The tube actor interacts with the system in two directions.

- The image on the tube is captured by the camera (15).
- The image is composed on the screen by tube supplies (16).

In Table 3 an overview of the identified use-cases is given. The complete use-cases are given in appendix F.

Use-case	Comment
1	Login
2	Select tube file
3	Choose measurement
4	Choose field
5	Choose colour
6	Execute video search
7	Execute colour recognition
8	Execute basic measurements
9	Execute basic framegrabber functions
10	Change image type
11	Select data collection system
12	Add new user
13	Edit parameters and variables
14	View measurements
15	Capture image
16	Compose image on measured object (tube)

Table 3 : Use-case of actors

7.1.10. Context diagram

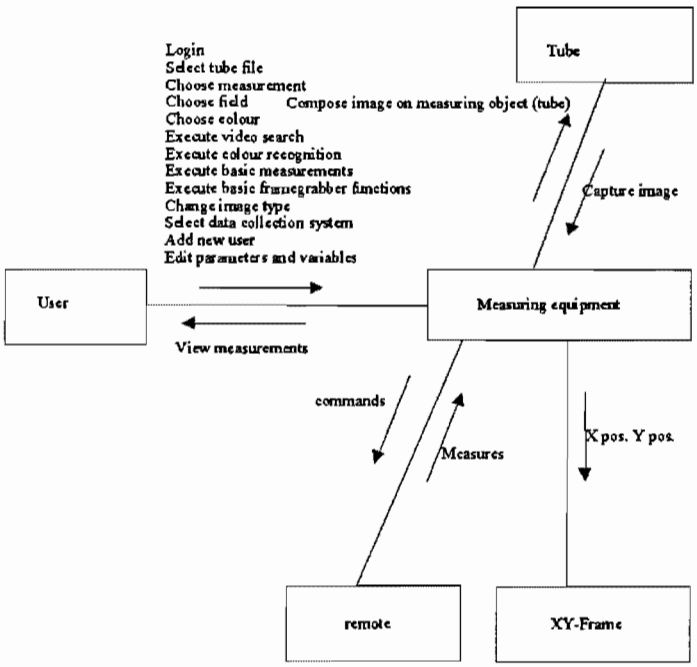


Figure 2 : Context diagram

With the definition of the major objects and actors, and the determination of the fundamental communications from and to the actors, the system scope has been outlined.
The system scope, captured in a context diagram is seen in figure 2.

.

7.1.11. Problem statement

A textual description of the problem to be solved can be given as follows:

There are two types of users. The first one is the operator. The second one is the maintenance engineer. The system must know which user is using the system. Before the user starts he selects a tube file that corresponds with the CRT he is going to measure. In that tube file, CRT specific parameters are stored. The user is able to execute measurements on a measured object. In case of the demonstrator the measured object, as in all cases at the current time, is a cathode ray tube.
The demonstrator is a copy of the COPER T1 and the main measurements are: convergence, eccentricity and linewidth. These measurements are also base measurements for other MMM equipment.
CRT calibrations are very complex and the optimal calibration of the CRT is an average measured at several places on the screen. For measurements with COPER T1 the screen is divided in 5 x 5 – measuring fields. The user must be able to select one of these fields. For several measurements a specific colour, picture or location is needed, so the user must be able to make a selection. If the COPER T1 is not equipped with an XY-frame the camera is placed

by hand. This is very inaccurate so a video search must be available. In the current MMM equipment CCD-cameras and LDR's are used to measure the CRT. The CCD cameras (black and white) and the LDR's can only detect grey levels so the equipment is able to conduct a colour recognition cycle to determine which dot is of what colour. The images are composed on the CRT by tube supplies.
The main measurements all use elementary measurements to calculate the results. For error detection and engineering purposes these elementary measurements are available to execute. Due to the fact that image processing is the basis on which the equipment works a frame grabber or other vision card is used to process the images received from cameras or LDR's. For evaluation and engineering purposes the basic functions of these cards must be available. Currently in MMM equipment, SBIP cards are used. The functions of currently implemented SBIP cards can be found in the Users guide [Mullekom 1996].
For manual measurements it must be possible to select different images. If a user has maintenance privileges he is able to edit program parameters, tube parameters, position parameters, colour recognition parameters, eccentricity parameters, convergence parameters and linewidth parameters. With those privileges he is also able to enter new users to the system.
The user must see the measurements made to judge the quality of the CRT.

7.2. Domain analysis

7.2.1. Candidate classes

The standard OMT approach to finding the domain classes is to examine the problem statement for candidate classes. Typically, nouns, pronouns, noun phrases, and implied nouns are underlined for further evaluation. To gain a complete list of candidate classes the user manual of the existing COPER T1 is also evaluated. In appendix G a complete list of candidate classes is available. Trivially unsuitable candidate classes are not included in the list. A filtered list of candidate classes obtained in the list from appendix G is shown in table 4.

Video search	Tube
Colour recognition	Field
Convergence	Picture
Eccentricity	User
Line width	Logging
Measuring equipment	Remote control
Sbip	
Camera	
XY-Frame	
Tube supplies	

Table 4 : Candidate classes

7.2.2. Model dictionary

Now a semi-formalised textual description of the candidate classes can be given. That list is placed in appendix H. At this stage the tool rational rose is used to develop the demonstrator application. The classes found in the previous section are entered into the class diagram of rational rose. Because of the big amount of classes only a few classes are shown in figure 3 which shows the input screen of rational rose. During the proceedings of this document only a part of the classes from the demonstrator will be shown. As an example of how rational rose works the complete input screens are shown. The final release with all classes is available in appendix N.

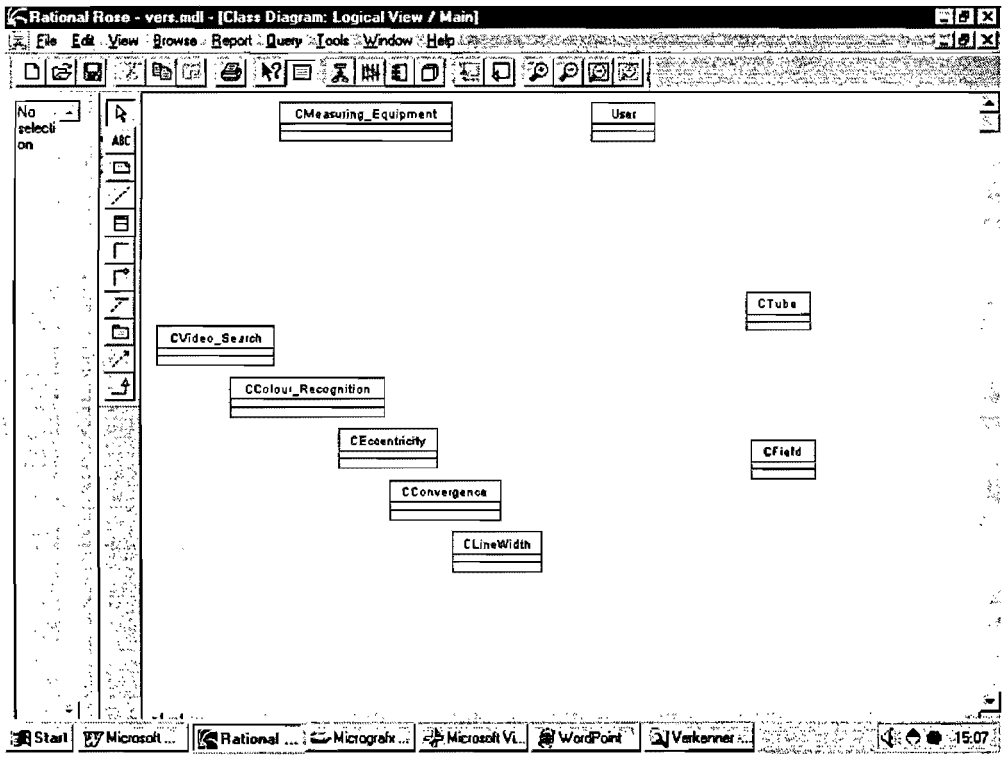


Figure 3 : Rational Rose class input

7.2.3. Associations

In Rumbaugh's book [Rumbaugh, 1991] the definitions of association, aggregation and generalisation are explained. Looking at every pair of classes on the candidate class list all associations in the demonstrator model can be found. In table 5 the associations are drawn.

Video search	*											X				X
Colour recognition		*										X				X
Convergence			*									X				X
Eccentricity				*								X				X
Linewidth					*							X				X
Measuring equipment	X	X	X	X	X	*	X	X	X	X	X			X	X	X
Sbip							*									X
Camera								*	X							X
XY-Frame									*							
Tube supplies										*						
Tube										X	*	X				
Field												*	X			
Picture													*			X
User														*		
Logging															*	
Remote control																*

Table 5 : Associations

note: * has an association with X

To obtain a better design the measurements are separated in two parts. First the measurement itself (how it is executed), secondly the parameters used to execute the measurements, and the measured values. The first is associated to the measuring equipment and the second is associated with the fields of the system. Table 6 shows the four extra classes added to the list of candidate classes.

Colour Recognition parameters	Eccentricity parameters
Convergence parameters	LineWidth parameters

Table 6 : Extra classes

After adding the new classes and associations in *Rational Rose* the input screen looks like figure 4.
The multiplicity of an association is the number of instances that participate in the association. Likewise the association has a role at each end of the association. For clearness only, the aggregation between class measuring equipment and class tube is added with roles and aggregation name. Aggregation is stronger form of association. It shows the relationship of a

whole to its parts. The multiplicity is 1 to 1, in which case nothing has to be added. For the aggregation between class tube and field the multiplicity is 1 to 25.

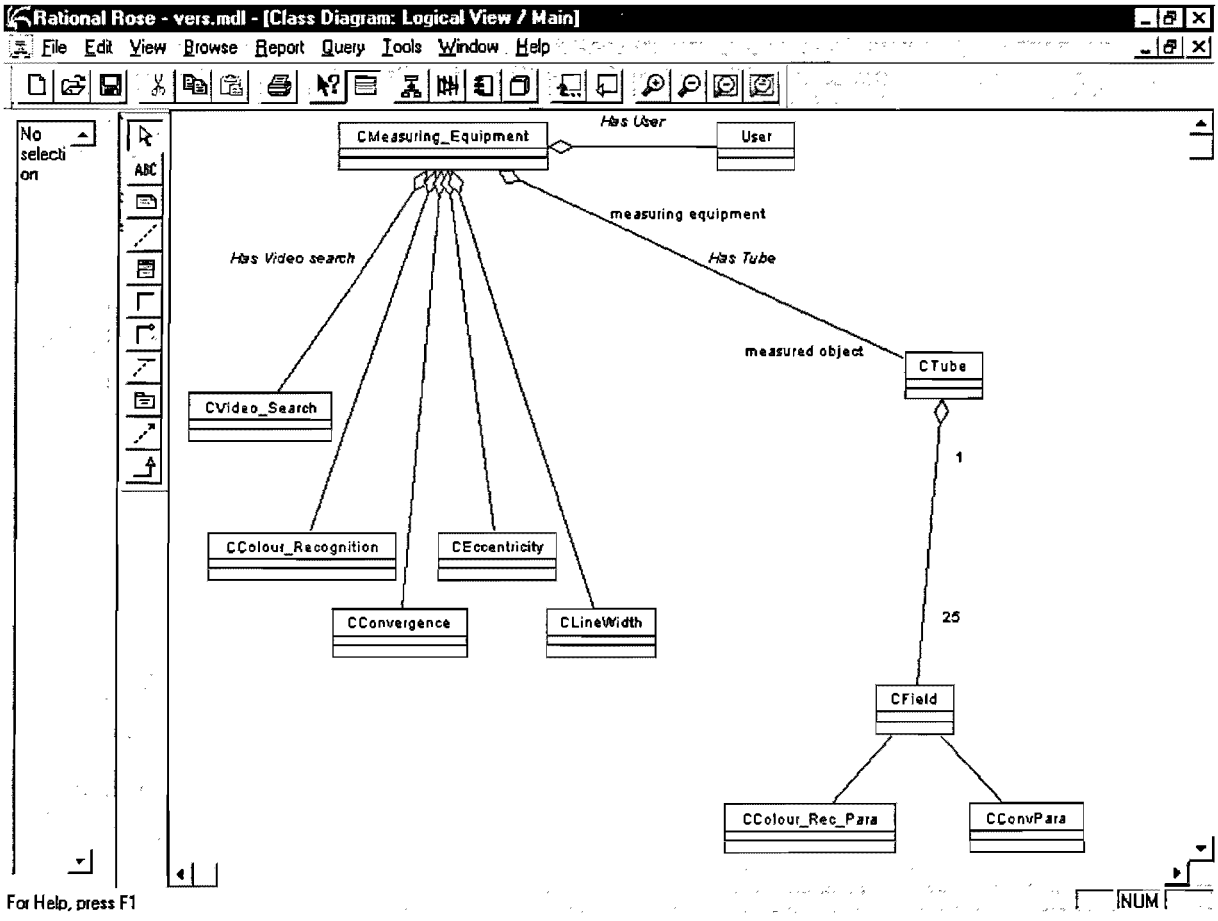


Figure 4 :Rational Rose with classes and associaties

7.2.4. Attributes and Operations

To distinguish an object from others and further define the roles in which the objects participate, essential settings and behaviours must be identified. There are two types of features that need to be identified, (OMT) attributes and operations. Attributes are the things that the object is responsible to know, while operations are the behaviours that the object is responsible for performing. At this stage it is important not to analyse in great detail because one might loose overview.

7.2.4.1. Attributes

The attributes for the demonstrator are found in the problem statement and in the user manual for the COPER [Mullekom, 1996]. A list of the attributes with the owner classes is found in the model dictionary, which can bee seen in appendix H. To follow the tool input and to clarify the demonstrator design, the tube class is worked out in detail.

The tube class represents the measured object. In the user manual of the COPER T1 one can read that it is possible to edit the general tube parameters. These parameters are specific for one kind of CRT. The tube object is responsible to know these parameters so they are added to the attributes of the tube class. The problem statement does not add more attributes. The attributes found for the tube class are listed in table 7.

m_TubeName	m_DeltaCentreX	m_1/2VertFieldOfView
m_Type	m_DeltaCentreY	m_LineFrequency
m_DeltaThreshold	m_VideoShiftStepSizeX	m_FrameFrequency
m_DeltaGain	m_NumberOfVideoShiftSteps	m_ClockPulseTube
m_VideoSensitivityFactorX	m_PulseWidth	
m_VideoSensitivityFactorY	m_1/2HorzFieldOfView	

Table 7 : Attributes of tube class

Following a style guide, attributes are started with m for member and the first letter of each additional word is capitalised.
The demonstrator is designed in visual C++. In that language there are several ways to define the attributes. The most common are: Public, Protected and Private. To preserve data hiding, and to make sure the class is the only responsible for the attributes the attributes for the tube class are declared private. The tool *Rational Rose* will automatically produce (if desired) public access functions.
Private declared attributes are only accessible by operations of the class itself.
Public declared operations are accessible by every class.

7.2.4.2. Operations

The tube class must be able to load and save its parameters. As one can see in figure 5 the parent of the field class (25 fields) is the tube class. In the field class an attribute m_Selected is available to know if the field is selected. If an object wants to know what field is selected he asks it to the tube class because it knows how to access the fields. Therefore a third operation is added to the tube object. It might be possible that more fields are selected at the same time so a fourth operation is added. The four operations are written in table 8.

m_Save(filename)	m_GetSelectedField ()
m_Load(filename)	m_GetNextSelectedField()

Table 8 : Operations in the tube class

These operations must be public because they are initiated from the parent class.

7.2.5. Rational Rose

After entering the attributes and operations in the tool *Rational Rose* the input screen looks like figure 5.

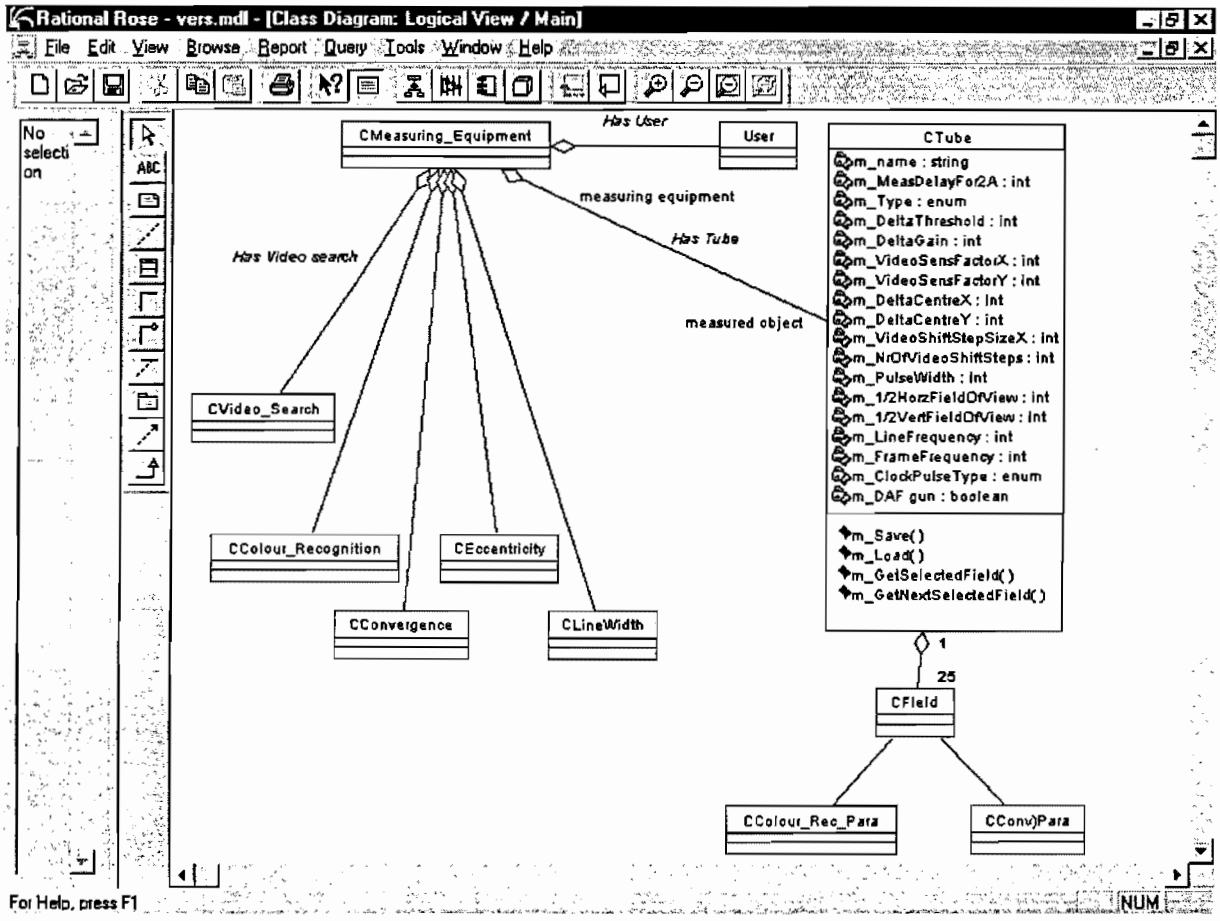


Figure 5: Rational Rose with attributes and operations

The lock in front of the attributes indicates that it is a private attribute. At this stage one can fill in all possible knowledge about the attributes and the operations. The input screen for the attribute m_LineFrequency is given in figure 6.

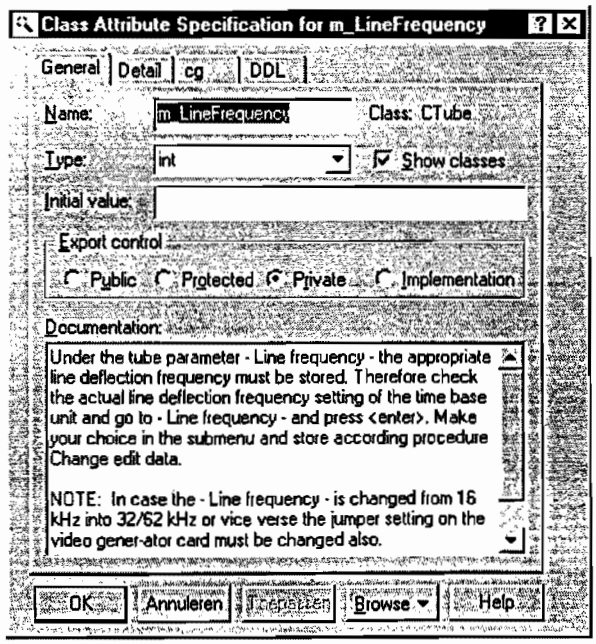

Class Attribute Specification for m_LineFrequency
General
Detail
cg
DDL
Name:
m_LineFrequency
Class: CTube
Type:
int
Show classes
Initial value:
Export control
Public
Protected
Private
Implementation
Documentation:
Under the tube parameter - Line frequency - the appropriate line deflection frequency must be stored. Therefore check the actual line deflection frequency setting of the time base unit and go to - Line frequency - and press <enter>. Make your choice in the submenu and store according procedure Change edit data.
NOTE: In case the - Line frequency - is changed from 16 kHz into 32/62 kHz or vice verse the jumper setting on the video gener-ator card must be changed also.
OK
Annuleren
Toepassen
Browse
Help

Figure 5 : Rational Rose attributes input dialog

7.3. Generalisation

As analysis progresses it becomes necessary to organise and simplify the model. Though it can be performed at any time, this process is usually delayed until enough details are found to detect commonalties.

In object-oriented systems, besides association and aggregation, there is a special relationship that may exist among classes. This relationship, called generalisation, is used as an organised sharing (or reusing) of features among the classes. A class that defines the common structure (attributes), behaviour (operations), relationships (associations), and meaning for a set of classes, is the generalisation of this set of classes. This process of finding the common abstraction and making a superclass, is done by examining the found classes.
Super classes or better base classes are also used to describe the interface to sub classes. When using this kind of interface design, it is possible to add classes at a later time. The system knows the base class so it knows how to communicate with it. Newly designed classes must be derived from the bass class.

The latter is important for the design of the demonstrator. The demonstrator is designed to perform the tasks of a COPER T1. But the design must also be able to perform tasks of other MMM equipment, if possible even without recompiling and linking. The measurements used in the analysis for the demonstrator are also basic measurements for MMM equipment. The class Method is added as a base class to the classes Video search, Colour recognition, Convergence, Eccentricity and Linewidth. The class Measurement parameters is added as a base class for the classes Colour Recognition parameters, Convergence parameters, Eccentricity parameters, and LineWidth parameters. If in the future a new measurement is necessary, or another MMM equipment must be designed, two new classes may be derived from the base classes and added to the system. In the future it might also be necessary that the measuring equipment is used for other measured objects, therefore the measured object Tube is added as a derived of a new base class Measured Object. As mentioned before the currently used image processing card is the Sbip, working in a VMS environment. But in the future it might be possible that a new processing card is available or the entire system is to be PC-based. Maybe new algorithms are designed to process the image. In all those cases a super class representing the interface to the card object makes it possible to adapt the system in a short time, to make the system even more flexible for newly or redesignable objects. The measuring dependent subsystems are added as inheritors of the super class used systems. In case of the demonstrator a Vision system is needed to obtain the images from the CRT. A Video system is needed to keep track of the characteristics of the images on the CRT. Finally a Supplies system is needed to drive the CRT to produce the image on the screen. Some MMM equipment also performs calibration operations like magnetising. For this equipment, a magnetising system may be added to the measuring equipment, together with the needed measurement classes.

Five new super classes are added to the design. The classes Method and Measure parameters describe the interface between measurements and the system. The class Measured object describes the interface between the measured object and the system. The class Used systems makes it possible to add new subsystems needed for other measurements. The class Video processor forms the interface between the system and the imaging processing part.

One note must be made. The design thus far is certainly not the final design. It is very well possible that subsystems derived from the super classes are complexes of other classes.
For the demonstrator all measurements and actions take place via the remote control. The classes as they are derived now directly place their requests to the remote control while in a fully operational system some classes must be more refined to fulfil the proper actions. As an example: the hardware design is not available at the current state of the design process. Especially the classes derived from the used System class are built with that design.

7.4. Application analysis

Application analysis use modelling techniques similar to those used in domain analysis, but now it includes those classes that represent the interface between the application system and the user.
The demonstrator is designed with the use of the user manual of the current COPER T1. This means that classes like the Video Processor are already in the design. Normally they are added in the application analysis.
A way to begin application analysis is to take each use-case in turn and define the details of the interaction between the actor and the system. The recommended way by OMT is to start with a scenario, that is a specific sequence of events being exchanged. There are two types of Scenario's in OMT. An external scenario is a scenario where the events between the actors and the system are modeled. An internal scenario also shows the events between the objects in the system.

7.4.1. Scenario

In OMT the external scenarios are presented by a numbered list of events. The following list represents the external scenarios for the demonstrator.

1. Login
   1. User enters name
   2. User enters password
   3. System looks for valid combination and starts the measuring system with the privileges of the current user.
2. Select tube file
   1. User selects file input screen.
   2. System shows file input name
   3. User selects tube file.
   4. System loads the selected tube file.
3. Execute measurement in single mode

(Scenario may be executed for linewidth, convergence, eccentricity and all other newly to add measurements)

   1. User selects measurement.
   2. System makes the selected measurement active
   3. User selects field
   4. System makes field active and starts selected measurement on selected field.
   5. System shows measurements.
4. Execute measurement in multi mode

(Scenario may be executed for linewidth, convergence, eccentricity and all other newly to add measurements)

   1. User selects fields for next measurement.
   2. System makes selected field active if it currently is not. And the system makes the field inactive if it currently is active. (1 and 2 are repeated until user satisfied with selection)
   3. User selects measurement.
   4. System measures the selected fields.
   5. System shows measurements.

For the measurements convergence and eccentricity the system asks the user to remeasure (complete colour calibration) before it starts the measurements.

5. Edit data
(Scenario may be executed for all possible data available)
1. User selects edit data input.
2. System shows edit data input window.
3. User selects the group of data to edit.
4. System shows the selected group input window.
5. User selects the data to edit .
6. System makes it possible to edit data.
7. User confirms or refuses newly entered data.
8. System makes data changes according to the confirmation.

6. Video control
(Scenario may be executed for Normalise video, Video search, Video move, Video colour, Video picture, Colour recognition)
1. User selects video function.
2. System shows input window for desired control function.
3. User enters new video properties.
4. User confirms or refuses newly entered data.
5. System makes video changes according to the confirmation

7. SBIP Functions with field selection in single selection mode
(Scenario may be executed for set gain, auto gain, auto threshold, mask in plane, pixel calibration show planes, histogram)
1. User selects one of the above mentioned SBIP functions.
2. System activates the selected function.
3. User selects field to be used in relation with the SBIP functions.
4. System performs function.
5. System shows measurements if any.

8. SBIP Functions with field selection in multi selection mode
(Scenario may be executed for set gain, auto gain, auto threshold, mask in plane, pixel calibration show planes, histogram)
1. User selects field to be used in relation with the SBIP functions.
2. System makes selected field active if it currently is not. The system makes the field inactive if it currently is active. (1 and 2 are repeated until user is satisfied with the selection)
3. User selects one of the above mentioned SBIP functions.
4. System performs selected function.
5. System shows measurements if any.

9. SBIP Functions with field selection in single selection mode
(Scenario may be executed for boot SBIP, Initialise SBIP Live video, Fixed video.)
1. User selects one of the above mentioned SBIP functions.
2. System activates the selected functions.
3. User selects field to be used in relation with the SBIP functions.
4. System performs function.
5. System shows measurements if any.

10. SBIP Functions with field selection in multi selection mode
(Scenario may be executed for boot SBIP, Initialise SBIP Live video, Fixed video)

1. User selects field to be used in relation with the SBIP functions.
2. System makes selected field active if it currently is not and the system makes the field inactive if it currently is active. (1 and 2 are repeated until user satisfied with selection)
3. User selects one of the above mentioned SBIP functions.
4. System performs selected function.
5. System shows measurements if any.

Before starting the selected functions the system asks the following function dependent data:

Set Gain	Gain,Threshold
Auto Gain	Gain, Delta gain, Threshold
Auto Threshold	Delta threshold
Mask in plane	Threshold white

For the function Mask in plane the currently selected colour is used. It may be changed in the Video control window.

11. Measurement utilities in single selection mode
(Scenario may be executed for Convergence Gravity XY, Gravity X, Gravity Y, Brightness, Brightness colour, Linewidth)

1. User selects measurement utility.
2. System activates the selected measurement utility.
3. User selects field.
4. System performs selected measurement.
5. System shows measurements if any.

12. Measurement utilities in multi selection mode
(Scenario may be executed for Convergence Gravity XY, Gravity X, Gravity Y, Brightness, Brigtness colour, Linewidth)

1. User selects field to be used in relation with the utility functions.
2. System makes selected field active if it currently is not and the system makes the field inactive if it currently is active. (1 and 2 are repeated until user satisfied with selection)
3. User selects measurement utility.
4. System performs selected measurement.
5. System shows measurements if any.

Before starting the utility linewidth the system asks the user to enter the focus direction and the level.
For the functions linewidth and brightness colour the currently selected colour is used. It may be changed in the Video control window.

Scenarios can also be depicted graphically, using a message trace diagram to capture the pattern of interactions for each one. In the message trace diagram, columns are used to indicate the participating objects in the scenario, usually with the actor(s) to the left. A message trace diagram shows the events and messages passing among the objects by the use of ordered labeled arrows. The internal scenarios are given in appendix O.

7.5. System design

After determining the system's requirements, the overall approach to the solution, its system architecture and style must be determined. For large systems this is normally the stage at which a lot of system organisation is handled. For the demonstrator (a one-man design) however this stage only comprises partitioning the system into subsystems and allocating subsystems to components (e.g. hardware, software, and operations).

Subsystems are typically chosen in a way that organises the systems into distinct layers and partitions. For the demonstrator three layers are chosen. On top there is the user interface. In the middle the task dependent layer is situated and at the bottom the hardware dependent layer is found. One can identify more subsystems but for the demonstrator these three are sufficient. For future use and increasing reusability more subsystems are a must. The hardware layer is not further explored. The demonstrator as implemented performs the measurements by accessing another COPER T1 with a remote link. No hardware dependent code is necessary at this stage. In [lambert, 1996] a MMM equipment is designed using OO in combination with visual C++. In that design the subclass 'hardware platform' can be identified. The design is very solid and it forms a great blueprint for a lot of MMM equipment. At this time all major MMM equipment is based on VME platforms in combination with PC based platforms. For the interaction with the users a great variety of input/output devices are available. In the future one likes to (if possible) go to one uniform user interface. (This study is part of a larger study to investigate and migrate to that vision). Older MMM equipment is normally not replaced by a newer one's but most of the time it is updated. The hardware layer in the first steps of the migration forms the connection between PC and VME systems. In later times or for new equipment it is necessary to develop PC based hardware. (Only if a fully PC based platform is desirable).
The policy for MMM is to avoid risk as much as possible. Therefore they use proven development environments and only change if another has proven to be stable. In [lambert, 1996] Windows version 3.1x was used and the newest relevant developing environment was MSVC 1.51. In the future they want to migrate to (if proven stable) Windows NT. Currently they are investigating newer development environments but until a proper decision is made MSVC 1.51 is the choice to make. The demonstrator will run on a stand-alone PC but if the hardware subclass is implemented and the proper measurement code is added it can run on a VME-PC/PC.

The subsystem user interface contains all objects that interact with the users. These objects accept the events of the users. The user interface contains classes that are added in the design phase. Because of the visual capabilities of the development environment used for the demonstrator and because of the lack of knowledge of its capabilities the user interface classes are designed with the aid of the application studio. The user interface must comply with the design rules found in chapter 4.

7.5.1. Code generation

For the demonstrator this is the last stage in which the development tool is used. The design at this point is converted to code by the code-generator of RATIONAL ROSE. The default property sets are used to generate the code for the demonstrator. In appendix J the .H file of the generated code for the tube class is shown. And in appendix K the .C file is shown.
In those files one can see that there are a lot of inline statements. This is one of the switches in the property set. Inline code is faster in execution because every declaration of the inline function is replaced by the code itself.

7.5.2. Classes

7.5.2.1.CMeasuringEquipment

The class *CMeasuringEquipment* is the main class of the demonstrator. It contains the associations to the measurements, the used systems, the measured object (tube), the user, and especially for the demonstrator the remote control. Further it manages several user interactions. In the windows environment the common way to select items is by the menu system. But for a user interface with touch screen input, the screen dimensions needed to design a menu system would be enormous. Another argument not to use the standard menu structure is the performance gain when using symbols and labels for selection purposes. See Chapter 4 for more information about touch screen constraints on user interfaces. Selecting items is done by specially designed dialogs with push buttons to select the item.

For the first version of the demonstrator the associations to the methods of the measurements are implemented statically. But for reusability and flexibility these associations must be implemented dynamically. The available methods should be detected during program start,. as mentioned in previous sections. All classes are developed to be able to use the dynamic linking. Because of this dynamic nature on the amount and kind of measurements, the user dialog for the available measurements is controlled in the *CMeasuringEquipment* class.

The current COPER T1 has grouped several elementary functions for analysing/development purposes. This group contains functions provided by several different classes. To make a proper design the initiations of those functions are handled by a user dialog controlled in the *CMeasuringEquipment* class and visualised as 'Meas. utilities'.

For the same reason as mentioned above the dialog to handle video operations is controlled by this class.

The dialog that gives access to the various parameters distributed over several other classes is also controlled in this class. This dialog is dynamic because the measurements also have parameters that can be changed. The actual dialog that provides the parameter manipulations for the dynamic classes is controlled by the classes itself.

To avoid distraction the operator only sees the dialog with the available measurements. But for analysing, debugging and system set up more dialogs must be available. Displaying available dialogs is handled by an extra dialog. The *CMeasuringEquipment* class controls the extra dialog.

As can be seen in the massage trees printed in appendix the *CMeasuringEquipment* object is responsible for the execution of the next action. It depends on the execution mode if a selected measurement is to be executed immediately or it has to wait until a field is selected. To tackle this problem three functions are added. The first defines that the next action to be taken is the selected one. And the second executes the selected action. If the execution mode is Multiple selection, the second function is called immediately. If this is not the case, execution is started if a field is selected. Calling the third function that executes the selected action does this.

See appendix P for the user interface layout of the previous mentioned dialogs.

7.5.2.2. CMeasuredObject

The Class *CMeasuredObject* represents the measured object. For now the tube is the only object to measure, but in the future there may be other objects so the class CTube is derived from the class CMeasuredObject. The CRT is divided in 25 fields. The only time for a field to be visual on the user interface is for selection. To avoid a lot of programming and make it possible that every field displays itself (the OO way), the dialog of the tube class is also responsible for the visualisation of the fields. If a field is selected this is passed to the appropriate field object.

7.5.2.3.CTube

The parameters for the CRT are situated in this class so the dialog to edit those parameters is also controlled from within this class.

The measurement parameters are associated with the fields. In multi selection mode more than one field can be selected. The parameters that are the same in all those fields must be displayed in the measurement parameter dialog. Or if a change is made it must be changed in all selected fields. The locations of those parameters must be available to the dialogs. To overcome this problem the request to do a parameter update of the measurements is situated in the tube object. This object knows the available fields and knows the locations of the available measurement parameters. It is neither wanted nor necessary that this object has knowledge about the interior of the measurement object (dynamic). If a update request is done (single or multi mode) The function that executes the dialog request, sends an array with pointers of type *CMeasureParameter*. This is the base class. If field X is not selected the array location X is filled with a NULL pointer. The array is sent to a arbitrary measure parameter object of the right type.

For instance, if the user wants to change the convergence parameters of the selected field, the type to send to is of type convergence. Once again this is not hard coded. The type is passed as an argument to a function in the field object.

Two functions are available to load and save the CRT dependent parameters. The argument of these two functions is the filename. The two functions work independent of the file contents. The arrangements of the parameters are solely determined by this function, and therefore object independent. Another measured object most probably has other parameters. So the implementation can not be inherited. But the function must be implemented for the system so it is declared virtual in the measurement base class. In appendix Q a part of the file is given as an example.
See appendix P for the user interface layout of the previously mentioned dialogs.

7.5.2.4. CField

As with most other parameter dependent classes the dialog for the field dependent parameters is controlled in its own class. As mentioned earlier, this is the object in which the measurement parameter objects are linked. For the first development the objects are linked statically. This means there is a variable for every measurement parameter. To fulfil the needs this must be changed to a dynamic construct or to an association of a linked list of measurement parameters. In the last case the name of the measurement is passed as an argument. By passing the name of the desired measurement every member of the linked list can determine if it must take the appropriate action. In the first release 5 pointers are implemented. The first four are the parameters for the measurements: colour recognition, convergence, eccentricity and linewidth. The fifth is added to store the field related parameters and measurements of the 'meas. utility' dialog. In figure 7 the currently implemented version is shown. In figure 8 the desired implementation is shown.

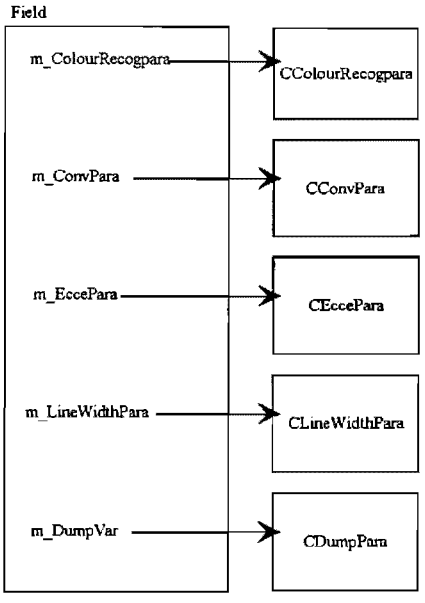


Figure 6 : Actual implementation

As with the tube class two functions are available to load and save the tube dependent parameters. The argument to these two functions is the filename.

After executing a measurement it is visualised in the tube dialog. The involved measuring object initialises a measurement, and it is not desired that those objects write directly in the tube object. The field object is the place to store a flag to the last updated parameter object. Using this construct the tube class itself is able to visualise the last measurements. Some measurements have multiple values which can not been displayed at the same time in the tube dialog. The indication of which value to show, and how to get this value is handled in the involved object.

See appendix P for the user interface layout of the previous mentioned dialog.

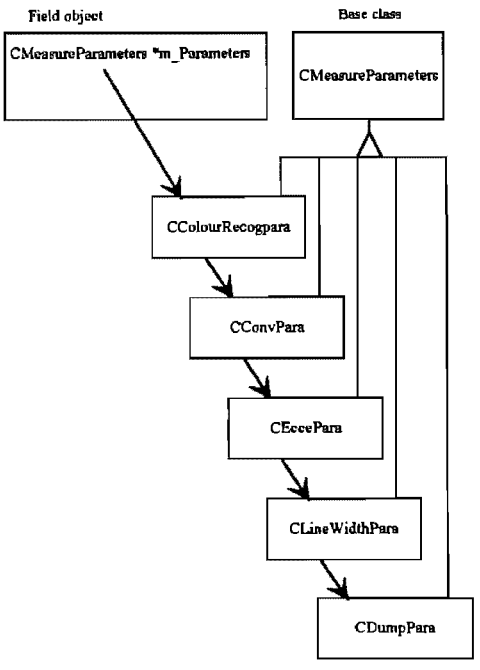


Figure 7 : Future implementation

7.5.2.5. CColourRecogpara, CConvPara, CEccePara, CLineWidthPara, CDumpVar

These classes contain the measurement parameters and measured values. These values differ per class but the total structure of the class shows a lot of comparisons in structure. They all have a function that handles the user dialog for changing and viewing the data. They have load and save functions. As mentioned in the previous chapter, the value to be displayed in the tube dialog after a measure can be received by calling one function which automatically returns the desired value. The value to return (if more possibilities are possible) can be set in the user dialog of this class.

The load and save functions are the same as in the other classes. The Microsoft Foundation Classes (MSFC) also contains a base function to store the data of a class. The only problem is the fact that a generated file is unreadable for a human user. By designing a new load and save function this problem is solved. Although it is a frequently used function the implementation differs per class because of the difference in data.

The dialog handling of these classes is much more interesting. As mentioned in section 7.5.2.3. the function that handles the request to show the dialog, is accompanied by an attribute that contains pointers to classes of the same type. The pointer is NULL if the referenced field is not selected. During development a second attribute is added to indicate that a dialog may be visualised without the need of an already opened dialog. This feature is added to avoid the need to control the handling of the dialog in another object. In this way more objects can initiate a request to show data in the dialog if it was open and do nothing if it was closed. Now depending on the second attribute three actions may take place: a dialog is opened, the contents of an existing dialog is changed, or nothing happens.

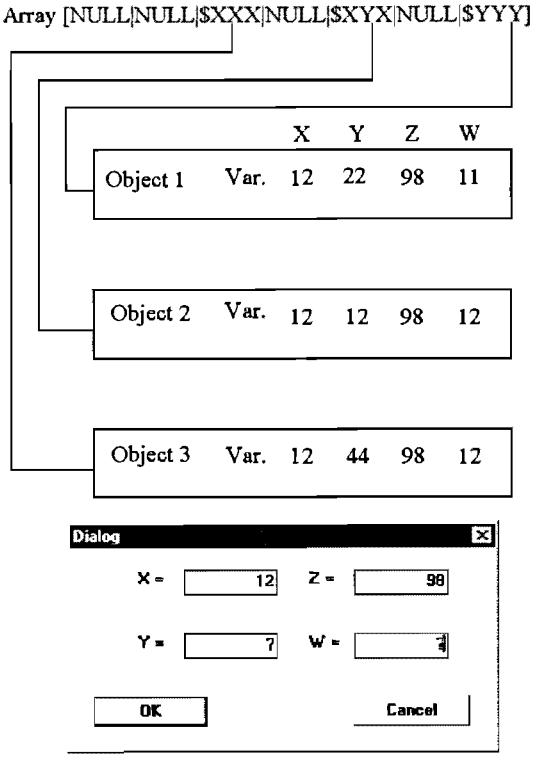


Figure 8 : Update mechanisme

It depends on the contents of the values in the objects what is going to be displayed.
E.g. three pointers in the array are not NULL meaning show the data of the objects where the pointer points to. The first step is to compare the data in those three objects. If a value is the same in all objects that value is shown. If one differs an 'unknown' sign (question mark) is shown. In figure 9 this example is shown graphically.

The previous can be done for one (no comparison is necessary) to many objects. The values to display are stored. If the dialog is already shown, a comparison is made between the new values in that dialog and the old values stored in the same manner as just mentioned. If they differ the user has made changes. If this is the case, the user is asked to save or cancel the new

data. The latter is done by a separate function because it can also be used if the dialog is terminated. If a save action is desired the previous selected objects are updated. Trivially a storage array of pointers is necessary to keep track of the previous selected parameter objects. After these actions the newly found data is given to the dialog. If no dialog was open and it is allowed, a dialog is opened. Checking for a change by the user is done before comparing the new data variables in the objects. In this way only one mirror of data variables is necessary. Another possibility is to create a new object of the same type and store the data in that temporary object.

A note is to be made. The object that creates the pointer array uses pointers of type *CMeasures_Params*. Because that object is part of the system it does not know the type of the parameter class. But within the receiving object a C++ type cast is made to its own type because he must access the measurements and parameters. This is no problem because an object knows its own class.

New classes for other measurements must all contain the same structure. This can be guaranteed by deriving these classes and a new to design class from the base class *Cmeasureparameter*, defining the needed functions virtually. By making a more sophisticated design it is possible to make a set of utility functions that can be used to perform the desired operations on random variables.

By implementing the classes in the described way, a truly OO concept is used for viewing and editing the measurements values and parameters. The object is the only one responsible for displaying the contents. The objects that request such a view only have to send the appropriate pointer array of the base class for the parameters.

See appendix P for the user interface layout of the edit/view dialogs.

7.5.2.6. CVideoSearch, CColourrecog, CConvergence, CExcentricity, CLinewidth

The objects that contain the measurement methods are very simple for the implementation of the demonstrator. The demonstrator does not have to execute the real measurement code. Only a call, with the appropriate attributes, to the remote control object is sufficient to retrieve the desired measurement values. For the remote control line it is not possible to place more than one request. So to reduce overhead and excessive programming, the calling measurement object waits until the measurements are received. The retrieved values thus may be passed by the attributes in the call.
In the remote control link it is not possible to send the measurements specific parameters stored in the measurement parameters. For the time being there is no access necessary to those objects. But if the measurement codes are implemented in these objects (complete working equipment), calls to the parameter classes are necessary. A measurement object always knows the type of measurement parameter object, because they are designed together. But it does not know the location of the object. During start-up of the program a member function in the measurement method object initiates a call to the system to request for the appropriate pointer. The system then asks the field object to generate the array of field dependent measurement parameter object pointers. As mentioned before there are several ways to implement the dynamic linking. And the generation of those pointers depends on the implementation. If the

list of pointers is generated it is returned to the object that asked for it. Before performing a measurement the measurement method object asks the system which field is next to be measured. Accordingly a call to get the correct parameters is initiated. Parameters that are not field dependent are kept in its own object. By touching the button of the measurement twice in a short time, the dialog to edit these variables is shown. The measuring equipment object controls the measurement selection dialog, so that object requests the measurement method object to show the appropriate dialog. In case of the demonstrator only the linewidth object contains field independent parameters. The linewidth property dialog is shown in appendix P.

7.5.2.7. CVisionSystem

The vision system represents a collection of objects that is responsible for the collection of measurements. Not only the placement of the cameras is situated here but also the conversion from an image on the measured object to digital data is handled here. The basic calculations from data to measurement values are also situated here. The vision class itself is only a collection place. The real actions take place in the aggregation objects *CVideoProcessor* and *CCameraCabinet*.

7.5.2.8. CVideoProcessor.

Currently MMM uses a very powerful processor based on a single board image processor card built around the TMS 34010/34022 Graphics system processor. It contains instructions for pixel manipulations on video memory. The video memory can store an image supplied by a CCD camera. On the card a system memory contains the program for the processor.

In the future other image cards may be used. It might be possible that the image grabbing, and the image processing is split. Doing so creates a diversity of implementation choices. Due to the fast Pentium processors used in PC based industrial computer systems, the choice could be to use a frame grabber and process the images on the main processors itself (multiple processor systems are available in industrial computers). If the processing is so complex or time constraints are important special DSP cards can be used to process the images.

The base class *CVideoProcessor* provides a flexible basis in the future. The necessary operations are defined in the base class. In the future other classes can be derived from this class to provide the needed functionality. For the demonstrator using the SBIP, the SBIP class is immediately derived from the base class. The defined methods in the base class are a little coloured because the are retrieved by looking at the current capabilities of the SBIP card. Extra methods may be needed, but were not implemented because of the limitations of the SBIP card. For the demonstrator the available methods are sufficient.

7.5.2.9. CSBIP

For the demonstrator and currently all MMM equipment the card to use is the Sbip card. For video processing a lot of preferences are available. They are accessed by the dialog, controlled in the derived class from the base class *CVideoprocessor*. A different

implementation asks for a different dialog, in this way the system does not have to know anything from the available preferences.

The member functions are not implemented for the demonstrator because all measurements are handled by remote control access and in the demonstrator no SBIP is available. Actions initiated by the user in the dialog are interpreted by a member function and then redirected to the remote control object.

7.5.2.10. CCameraplacing

The task of the class *CCameraplacing* is to place the camera on the correct spot of the measured object. If a fully automated system is built more cameras or an XY table are available. If none of this gear is available, hand placing is required. The object of type *CCameraplacing* is responsible for the selection and positioning of the camera or cameras. In this object a member function takes as arguments the desired location on the measuring object. Now if the apparatus is automatic, that function will take actions to select or move the right camera. When this is not available this function will send a message to the user to move the camera to the right spot. This object is the only one responsible for the selection so it is aware of the current position of the camera.

7.5.2.11. CVideoSystem

The Object of type *CVideoSystem* is responsible for the picture to be shown on the measured object. For flexibility in the future the picture properties on the tube are separated for every field. By doing so it is possible to adjust all properties of the picture in a single field. Currently only the move property is changed separately. The others (Kind of picture and colour) are all updated at once. This is due to the fact that the video card that produces the image is unable to show different pictures with different colours on the screen. The changing of the picture properties is therefore handled in a single object with type *CVideoSystem*. In that object an array variable is defined of length 25. This array contains pointers to the objects that represent the picture properties in one field. The system is able to alter these properties by the object of type *CVideoSystem*. The user must alter these properties too. To avoid difficult programming the dialogs to alter position, kind of picture, and Colour are controlled by the same object of type *CVideoSystem*. This is the object that communicates with the remote control object to change the properties of the image on the remote controlled COPER T1. The dialogs can be seen in appendix P.

7.5.2.12. CPicture

The *CPicture* class is used to enter the 25 field dependent picture properties. This class only contains the three properties (colour, type, location) of the picture. By adding pointers to the Field class and to the Video class the pointers can be referenced during creation to point to the corresponding field and to the object controlling the dialogs. This class contains information about preferences during measurement so in this class the load and store member functions are used.

7.5.2.13. CSuppliesControl

The class *CSuppliesControl* is completely empty in the demonstrator design.

7.5.2.14. CUser

For touch screen based user interfaces, left handed persons need a left oriented user interface and right handed need a right oriented user interface. To optimise the performance of the user it is important that the user has a personal calibrated touch interpretation. For MMM there is a difference in normal operators and maintenance operators. As mentioned in chapter 2 this lies in the ability to change process parameters. All these properties are handled in the object of type *CUser*. The dialog is shown in appendix P.

7.5.2.15. CRemote

The currently operational COPER T1 is able to execute most of its functions by remote control. The demonstrator uses this remote control to obtain the measurements. All possible functions can be found in the user manual for the COPER T1 [Mullekom, 1996].
For the remote control class all functions possible with the remote control are implemented as member functions. Each of these member functions have the appropriate attributes to receive the parameters to measure with, and to return the measured values. The member functions call another member function to send the command to the serial port which is used to communicate with the COPER T1. The serial port is also implemented as an object of type *PORT*. This *PORT* class is developed by another section at EED.

7.5.2.16. General

To accomplish the functionality of the classes extra functions are developed. Those functions are not essential for the design of the demonstrator so they are not explained any further.

Because of the automatic generation of the blueprint for the demonstrator, and because the demonstrator is only a part of this study the C code is not revealed in this document. It is however possible to obtain the CD "touch screen feasibility" to look at the complete C code. The CD can be obtained by Philips EED section MMM.
The dialog classes are generated by making the view with Microsoft's application studio and then convert them to a class with the classwizard. The code for those classes can also be found on the CD.

8. Conclusions and Recommendations

8.1. Conclusions

The study reveals that users of MMM equipment are very divers. There is a great difference in education level and in the turnover rate of the users. Because of those properties, a menu structured dialog in combination with direct manipulation, is a very well suited dialog for users of MMM equipment. This dialog performs well with the aid of a touch screen IO. The developers of MMM can gain from this result because currently an update in the button panels means a new design. With touch screens only a new graphical user interface layout is necessary.

When designing a touch screen based application it is important to follow the constraints that are put on the design by the touch screen. A user-friendly system is necessary. By frustrating users because they can not touch a specific point easily, an antipathy against touch screen technology may be created. A new user has to take some time to learn the interaction between user and screen (approx. 20 min.). The new user must be positioned in front of the touch screen so only little parallax error occurs. After such a session the user is able to adapt the way he touches, if the angle of sight is not orthogonal to the screen.

The technology chosen (Surface acoustic wave) is shown to be stable in a simulated test environment. The communication between touch screen and computer must be protected at any time because a flash-over can kill the electronics (PC side).

For the design of the demonstrator, Rational Rose (OMT) is used as object oriented design tool and Visual C++ is used as developing environment. Basically this meant for the author: learning OO by scratch, using an unknown design tool and programming in a new environment. despite these problems, the results were very good. The visual aspects of C++ contribute to a nice graphical user interface without concerning too much about the underlying code. Rational Rose is a tool that closely matches with the literature about OMT design. The code generation as used was not optimal. But with the right settings a nice code body can be produced.

The design of the demonstrator in combination with the design of LabMag [Lamber, 1996] may be used as an indication on how MMM equipment can be designed in the future. Some of the classes designed for the demonstrator can easily be used by other designs. The latter is especially true for the classes in the user interface subsystem, like virtual Keyboards and input dialogs.

8.2. Recommendations

Due to time pressure an evaluation with end users is not available. With the current demonstrator this evaluation can take place in the future. The people which played with the demonstrator are very enthusiastic about its capability.

A real field test is not executed. Whether the touch screen technology is really factory resistant in not proven yet. This test should be the first to execute because if it fails it will be very hard to find a technology that can resist the harsh environment. A lot of people indicate that there is a lot of dust in the factory and that might effect the capabilities of the touch screen. According to the studies the dust should not be a problem for the chosen touch screen.

The demonstrator is developed with Visual C++ 1.51. This is very out-dated. New development platforms are tested by section MMM. The newer development platforms offer more functionality so if the MMI for MMM equipment is further developed, a study about the newly offered functionality can be used for touch screen applications.

references

Albert, A.E. (1982). The effect of graphic input devices on performance in a cursor positioning task. P*roceedings of the Human Factors Society 26th Annual Meeting*, 1982, pp 54-58.

Backs, R.W. & Walrath, L.C. (1987). Comparison of Horizontal and vertical Menu Formats. *Proceedings of the Human Factors Society 31 th Annual Meeting*, 1987, pp. 715-17.

Beringer D.B. and Bowman M.J. (1989). operator behavioral biases using high-resolution touch input devices. *Proceedings of the Human Factors Society 33 th Annual Meeting*, 1989, pp. 320-22.

Beringer D.B. and Peterson. J.G. (1985). Underlying behavioral parameters of the operation of touch-input devices: baises, models, and feedback. *Human Factors Society, 1985, pp. 445-58.*

Callahan, J. & Hopkins,D. (1987). An empirical comparison of pie vs. linear menus. *CHI '88*, pp 95-100.

Card, S.K. & Moran T.P. (1983). The Psychology of human computer interactions. (Hillsdale, N.J.:Lawrence Erlbaum Associats, 1983.

Card, S.K. (1981). The Model Human Processor: A model for making Engineering Calculations of human Performance. *Proceedings of the Human Factors Society 25th annual Meeting*, 1981, pp. 301-305.

Ezzell, B. (1992). Grafisch programmeren in windows 3.1, ISBN 90-229-3808-5

Foltz, P.W. (1988). Transfer between menu systems. *CHI '88 Proceedings*, May 1988, pp. 107-122, ACM.

Franey J.P. (1995). Field-Induced ESD from CRT's Its Cause and Cure. *IEEE Transactions On Components, Packaging, And Manufacturing Technology PART* A, vol 18 NO 2 June 1995

Franik, E.P. & Kane R.M. (1987). Optimizing visual search and cursor movement in pull-down menus. *Proceedings of the Human Factors Society 31 th Annual Meeting*, 1987, pp 722-26

Furnas, G.W. (1985). Experience with an adaptive indexing scheme. *CHI 85 Proceedings*, April 1985, pp. 131-36, ACM.

Galiz, W.O. (1989). Handbook of screen format design 3rd ed.(Wellesley, Mass,:QED information Science, Inc. 1989, pp. 173-77.

Good, M. (1985). The use of logging data in the design of a new text editor. *CHI 85 Proceedings*, April 1986, pp. 93-113, ACM.

Gould, J.D. (1988). Empirical Evaluation of Entry and Selection Methodes for Specifying Dates. *Proceedings of the Human Factors Society 32th Annual Meeting*, 1988, pp 279-83.

Granda R.E. & Teitelbaum, R.C. (1982). The effect of VDT command line location on data entry behavior. *Proceedings of the Human Factors Society 26th Annual Meeting*, 1982, pp 621-24.

W.D.Greason (1994). Analysis of Human Body Model for Electrostatic Discharge (ESD) wih Multiple Gharged Sources. *IEEE Transactions On industry Applications*. vol. 30 NO.3 May/June 1994

Greene, S.L. (1988). Entry based verses selection-based interaction methodes. *Proceedings of the Human Factors Society 32th Annual Meeting*, 1988, pp 284-87.

Hollands J.G. & Merikle P.M. (1987). menu Organization and user expertise in information search tasks. *Human Factors 29*, no. 5 (Oktober 1987), pp 577-89

Karat, J (1987). Evaluating user interface complexity. *Proceedings of the Human Factors Society 31th Annual Meeting*, 1987, pp 566-70.

Karat, J. & McDonald, E.J. (1986). A comparison of menu selection techniques: touch panel, mouse and keyboard. *Man-Machine Studies*, 25, pp,73-88.

Kraut, R.E. & Stephen J.H. (1983). Command use and interface design. *CHI 83 Proceedings*, pp. 120-24, ACM.

P.C.T. van der Laan: Electromagnetic Compatibility college dictaat 5785

Landauer T. K, Nachbar D.W. (1985). Selection from Alphabetic and Numeric Menu Trees Using a Touch Screen: Breadth, Depth and Width. *CHI '85 Proceedings*, April 1985, pp 73-78,

LedGard, H.F. (1980). The natural language of interactive systems. *Communications of the ACM*, 23 , pp. 556-63

Lee, E & Chao, G (1989). The increase utility of incorporation keywords in menu systems and users increase experimence. *Behavior & information technology*, 8, no.4 (July-August 1989), pp. 301-308.

Lockheed, M. (1996). Succeeding with the booch and omt methods. Addison-Wesley, ISBN 0-8053-2279-5

MacGregor, J. & Lee E. (1986). Optimizing the structure of database menu indexes: A decision model of menu search. *Human Factors*, 28, no. 4 (August 1086), 387-400.

Mayhew J.D. (1987). Principles and Guidelines in Software User Interface Design. Prentice-Hall, inc. ISBN 0-13-721929-6, 1987

McDonald, J.E. & Molander, M.E. (1988). Color-coding categories in menus. *CHI '88 Proceedings*, May 1988, 00 101-106, ACM.

McDonald J.E. & Stone J.D. (1983). Searching for items in menus. *Proceedings of the human factors society – 27th annual meeting*, 1983, pp 834-37.

Moran P, and Newell A, (1983). The Psychology of Human-Computer Interaction (Hillsdale, N.J.:Lawrence Erlbaum Associates, 1983.

Miller, D.P. (1981). The Depth/Breadth Tradeoff in Hierarchical Computer Menus. *Proceedings of the Human Factors Society 25th annual Meeting*, 1981, pp.. 296-300.

Mori, T. (1996). Analysis of ESD Immunity of Electronic Equipment Based on Ground Potentioal Variations. *IEICE Trans. Commun,*. Vol. E79-B. NO.4 April 1996

Mullekom, R. (1996). User manual software for copert T1. *Internal Report Philips 7322 055 2177.*

Norman K.L. Chin J.P. (1988). The effect of Tree Structure on Search in a Hierarchical Menu Selection System. *Behaviour & Information technology*, 7, no 1 (January-March 1988), pp 51-66

Paap K.R. & Hofstrand R. (1986). The Optimal Number of Menu Options per Panel. *Human Factors*, 28, no. 4 (Aug. 1986), pp. 377-86.

Potter, R.L. & Weldon, L. W. (1988). Improving the accuracy of touch screens: an experimental evaluation of three strategies. *CHI '88*, pp. 27-32.

Roberts, T.L. & Moran, T.P. (1982). Evaluation of text editors. proceedings, *Human Factors in computer-systems*, March 1982, pp. 136-41.

Rumbaugh, J. (1991). Object Oriented modeling and design. prentice-Hall, ISBN 0-13-630054-5.

Schelkunoff. S.A. The electromagnetic theory of coaxial transmission lines and cylindrical shields. *BSJT.* Vol 13, 1934

Schneiderman, B. (1982). The future of interactive systems and emergence of direct manipulation. *Behavior and information technology*, 1, pp. 237-56.

Sears A. and b. Shneiderman (1991). High precision touchscreens:Design strategies and comparison with a mouse. *International Journal of Man-Machine Interaction*, 34, 1991, pp 593-613

Snowberry K & Parkinson S.R. (1938). Computer Display Menus. *Ergonomics*, 26 (1983), pp 699-712

Shinar, D. & Stern, I.H. (1997). Alternative option selection methods in menu driven computer programs. *Human Factors*, 29, no. 4 (August 1987), pp. 453-60.

Somberg, B.L. (1987). A Comparison of rule- based and positionally constant arrangements of computer menu items. *CHI '87 proceedings*, May 1997, pp 255-260.

Stroustrup, B. (1997). The C++ Programming language third edition. ISBN 0-201-88954-4 Addison-Wesley

Teitelbaum, R.C. & Granada, R.E. (1983). The effects of positional constancy on searching menus for instructions. *CHI '83 Proceedings*, pp. 150-53.

Valk, M.A. (1985). An experiment to study touchscreen "burron" design. *Proceedings of the Human Factors Society 29h annual Meeting*, 1985, pp. 127-131.

Wallace, D. F., Anderson N. S. and Shneiderman B. (1987). Time stress Effects on Two Menu Selection Systems. *Proceedings of the Human Factors Society 31th annual Meeting*, 1987, pp. 727-31.

Wallace, D. F., Anderson N. S. and Shneiderman B. (1987). Time stress Effects on Two Menu Selection Systems. *Proceedings of the Human Factors Society 31h annual Meeting*, 1987, pp. 727-31.

Weisner, S.J. (1988). A touch-only user interface for a medical monitor. *Proceedings of the Human Factors Society 32 th Annual Meeting*, 1988, pp 435-39.

Whiteside, W. & Jones, S. (1985). User Performance with Command, Menu, and iconic Interfaces. *CHI '85 Proceedings*, April 1985, pp. 185-91.

Whitfield, D. and R.G. Ball (1983). Some comparisons of on-display and off-display touch input devices for interaction with computer generated displays. *Ergonomics*, 26, no.11, pp. 1033-1053.

Wijdeven, L. (1996). Labmag. Hogeschool Eindhoven 1996. Afstudeerverslag.

Wixon D. (1983). building a user-defined interface. *CHI 83 Proceedings*, December 1983, pp. 24-27, ACM

A. User Profile Checklist

Psychological Characteristics

Cognitive style:
Verbal/analytic
Spatial/intuitive

Attitude:
Positive
Neutral
Negative

Motivation:
High
Moderate
Low

Knowledge and Experience

Reading level:
No reading capabilities
Less than fifth grade
Fifth to twelfth grade
Above twelfth grade

Typing skill:
Non
Low
Medium
High

Education:
Elementary school
High school degree
College degree
Advanced degree

System experience:
Expert
Moderate
Novice

Task experience:
Novice in field
Moderate
Expert in field

Application experience:
No similar systems
One similar system
Some similar system

Native language:
English
German
French
Spanish
Taiwanese
Chinese

Use of other systems:
Little or none
Frequent

Computer literacy:
High
Moderate
Low
Non

B. Job and Task Characteristics

Frequency of use:
Low
Medium
High

Primary training:
None
Manual only
Elective formal
Mandatory formal

System use:
Mandatory
Discretionary

Job categories:
Operator
Production chief
Process specialist
Maintenance man

Turnover rate:
High
Moderate
Low

Other tools:
Telephone
Calculator
Adding machine
Other

Task importance:
High
Low

Task structure:
High
Moderate
Low

Physical Characteristics

Color-blind:
Yes
No

Handedness:
Right
Left
Ambidextrous

Gender:
Female
Male

C. Dialog Styles

A dialog style is an overall style of interaction. At least seven distinct dialog styles exist and are addressed here:

- Menus
- Fill-in forms
- Question and answers
- Command languages
- Function keys
- Direct manipulation
- Natural language

Most MMI's employ more than one dialog style, and each of these may not appear in its most pure form. In the following sections however a distinction is drawn to represent research findings, design principles and guidelines.

Menus

In menu-driven user interfaces, the primary form of interaction is a sequence in which the user is repetitively presented with sets of choices and asked to select one or more from among them. Choices may be presented as words in a list or as collections of icons.

Advantages and Disadvantages of menu dialog styles

Menus have particular advantages and disadvantages to other dialog styles. These are included in the following table.

Advantages	**Disadvantages**
Self-explanatory	Inefficient
Easy to learn	Inflexible
Few keystrokes	Impractical for numerous choices
Easy error handling	Take up screen "real estate"
Enhancements are visible	

Menus are most appropriate for:
USER PSYCHOLOGY
Negative attitude
Low motivation
KNOWLEDGE AND EXPERIENCE
Low typing skill
Little system experience
Low task experience
Low application experience
Frequent use of other systems
Low computer literacy
JOB AND TASK CHARACTERISTICS
Low frequency of use
Little or no training
Discretionary use
High turnover rate
Low task importance
High task structure

Design issues in menu systems can be divided into five areas:

- Menu structure
- Menu choice ordering
- Menu choice selection
- Menu invocation
- Menu navigation

Menu structure

One important issue regarding menu design is the number of choices and the number of levels to select from. There is a trade off between the two [Miller, 1981]. One extreme is to have only a few choices on any individual menu screen and, therefore, many or at least more levels in the menu hierarchy. An other extreme is to have many choices on a single menu screen. The problem is to find the optimum of user performance when navigating through a menu system to a target item at the lowest level.
According to Miller 8 items per menu screen was the optimum. He only uses 4 menu structures to retrieve this optimum:
Two choices per screen, six levels
Four choices per screen, three levels
Eight choices per screen, two levels
Sixty-four choices all on one screen, one level
The real optimum might lie between the Eight and Sixty-four choices.

Another study [Wallace, 1987] also investigated the depth versus breadth trade-off, and included some kind of time pressure. This study makes it clear that time pressure will affect performance. Interfaces to be used under such conditions should be tested under comparable conditions.

It is also important to organise within menu screens. Snowberry [Snowberry, 1983] evaluated the same depth-breadth conditions as Miller, but they organised the 54 items into particular groups within one screen. The performance was slightly better than the two level eight items per level structure.
Landauer [Landauer, 1985] studied depth versus breadth using a touch screen. in combination with structured menu items (numbers and letter ranges). This study indicated that maximum performance is reached if the number of selection items is between the 8 items on a screen with more levels and 64 items on a screen providing lower levels. Paap [Paap, 1986] found that optimum breadth values ranged from 16 to 78 items on one screen. That is, when menu choice items on a given screen can be ordered into meaningful groups. Collecting more information about depth breadth trade off [Lee & Chao, 1989] a good guideline for making the depth-breadth trade off in a menu hierarchy might be stated as follows. Choose a maximum breadth of 6 to 12 items per screen if no useful grouping can be imposed. Or if the menu items are complex of nature, chose a breadth of up to 25 if an organisation can be imposed to effectively group items and the items are not complex.
Menu orientation is also an aspect of menu structures. One study [Backs, 1987] investigated the horizontal and vertical orientation. Vertical menus performed faster but for better layout and compatibility with existing structures no generalisation can be made. A pie structured menu performed better for 8 items [Callahan, 1988]. However, some important qualifications must be considered. Larger amounts of items become impractical. It is hard to imagine how pie menus could effectively be presented in deeper menu structures. Finally pie menus require a graphic user interface.
It sometimes happens that some menu items may not be selected. To maintain the learning patterns navigating through the menu it is better to grey those items than to delete them [Somberg, 1987; Franik & Kane 1987] although for experienced uses deletion performs a little faster.

Menu choice ordering

The optimal ordering in user search and selection time is studied by several people. [McDonald & Molander, 1988; Hollands & Merikle, 1987; MacGregor & Lee, 1986].Given that menu systems are generally chosen to support novice, casual users, these studies suggest that ordering by logical semantic groups is a good organisation principle to employ. If the users however are high frequency more experienced users alphabetic ordering might be as usable as categorical ordering. Yet another choice of ordering is to group the most frequently used items[McDonald & Stone, 1983] A combination is probably the best way to obtain high performance. Note that in the McDonald & Molander study a touch screen was used to select the items. Searching for the desired item is one thing but selecting it is something else. It depends on the selecting device how long it takes to select the item.

Menu choice selection

A variety of selection mechanisms are available. The most common are single-character letter or number selection codes. Alternatives include moving a cursor through the list of choices with cursor control keys, or pointing directly to the choice with an alternative input device such as a mouse, joystick, or touch screen.
Selection codes provide faster selection times as compared to number codes or cursor selection, except for first time users in combination with a small number of options [Shinar & Stern, 1987]. When using selection codes clear label choice is very important. Cursor selection took more time because it depends on the item to select and the place of the cursor

how many keystrokes must be done. This drawback can be overcome with touch screen. Albert [Albert, 1982] compared a large set of pointing devices, including touch screen.

Menu navigation

Navigating in a menu system is facilitated by a number of design techniques. The most important guidelines are drawn from several research studies:

- Establish conventions for menu design and apply them consistently on all menu screens within a system.
- Use context labels, menu maps, and place markers as navigational aids in complex menu systems.
- If possible use direct access to menus and create macros to facilitate navigation for expert users.
- Think about backward navigation.

These guidelines are found in several studies[Teitelbaum & Granada, 1983; Foltz, 1988; Lee & Chao, 1989].

Fill in forms

A fill in form interface is similar to a paper fill in form. The only difference is the presentation, which is a computer screen.

Advantages and Disadvantages of Fill-in forms styles

Fill-in forms have particular advantages and disadvantages to other dialog styles. These are included in the following table.

Advantages	**Disadvantages**
Self-explanatory	Assumes knowledge of valid inputs
Efficient use of screen "real estate"	Assumes typing skill
Many possible input values	Error prone
Provides context	Knowledge of special keys
Enhancements are visible	Inflexible

Fill-in forms are most appropriate for:
USER PSYCHOLOGY
Negative or neutral attitude
Low or moderate motivation
KNOWLEDGE AND EXPERIENCE
Moderate to high typing skill
Little to moderate system experience
Little to high task experience
Low to moderate application experience
Moderate to frequent use of other systems
Moderate to high computer literacy
JOB AND TASK CHARACTERISTICS
Moderate to high frequency of use
Little or no training
Discretionary use
Low to moderate turnover rate
Moderate task importance
High task structure

The design of fill-in form systems can be divided into five separate issues:

- Fill-in form caption and field design
- Fill-in form input formats
- Fill-in form prompts and instructions
- Fill-in form navigation
- Fill-in form error handling

Fill-in form organization and layout

There are several areas to consider in the design of captions and fields. One study [Savage] compared four field indicators in combination with several cursor types. A conclusion that can be drawn from this study is that blinking block cursor in combination with broken underline field indicator is an optimal way to design input fields on a fill-in form. Note however that the performance study did not include actual speed or correctness of input.

Fill-in form input formats

To gain speed and reduce input errors the following design guidelines are appropriate:

- If possible provide completion of unambiguous partial input.
- Provide pop-up or pull-down menu's for fill-in fields with many entry options.
- Avoid complex entering fields.
- Break up long input fields.
- Provide defaults.
- Make high frequency inputs easy to express.
- Let the user specify the units of measurement.
- Allow abbreviated input.
- A system should be "case insensitive" when it really does not matter.
- Keep input fields short.
- Do not combine letters and numbers if possible.
- Do not require leading zeros.

These guidelines are drawn from several studies[Galiz, 1989; Gould, 1988; Greene, 1988].

Fill-in form prompts and instructions

Fill-in forms have prompts, or brief syntactic or semantic instructions associated with individual fields. Those are instructions for cursor movements, the use of special function keys, or screen acceptance and cancellation. The following guidelines apply:

- Provide prompts when use will be relatively infrequent.
- Prompts should be brief and unambiguous.
- Place prompts to the right of fields or in a MicroHelp line at the bottom of the screen.
- Provide instructions for navigation and completion.
- Place instructions in a consistent location across the screen.
- Use consistent terminology and consistent grammatical form.

Fill-in form navigation

The user must be able to move from field to field, to accept a filled in screen, to cancel a screen, and to move forward and backward between screens. The following guidelines can be applied:

- When a form is entered, position the cursor in the most likely default position.
- Arrange field groups consistently with default cursor movement.
- Allow forward and backward movements.
- Make protected areas on the screen inaccessible.
- Do not use auto tab unless fields have fixed lengths.
- Provide titles and page numbers.

Fill-in form error handling

Inevitably users will make errors entering data. To make those errors as small as possible the following guidelines can be drawn:

Allow character edits in fields.
Place the cursor in the error field after error detection.
Provide semantic and syntactic information in error messages.

Question and answer

As in a menu system the user is posed a single question. Like in fill-in form interface, however the user is expected to type in an answer.

Advantages and Disadvantages of Question and answer dialog style.

Question and answer dialog styles have particular advantages and disadvantages to other dialog styles. These are included in the following table.

Advantages	**Disadvantages**
Self-explanatory	Inefficient
Simple and nonintimidating	Assumes typing skill
Many possible input values	Error prone
Accommodates hierarchical task structure	Inflexible
Complete and clear prompting	No forward context
Enhancements are visible	

Questions and answers are most appropriate for:
USER PSYCHOLOGY
Negative attitude
Low motivation
KNOWLEDGE AND EXPERIENCE
Moderate to high typing skill
Little to moderate system experience
Low task experience
Low application experience
Moderate to frequent use of other systems
Low computer literacy
JOB AND TASK CHARACTERISTICS
Low frequency of use
Little or no training
Discretionary use
High turnover rate
Low task importance
High task structure

Because questions and answer interfaces are relatively rigid, structured dialog style, they are best suited to tasks that are themselves highly structured.
There is not much research on question and answer interfaces. The guidelines that follows are based on basic research and aspects of other dialog styles:

- Maintain system titles for navigating proposes.
- Use clear and simple languages.
- Provide brief prompts and instructions.
- Make a visual difference between questions, prompts instructions and user input.
- Minimise typing requirements.
- Allow flexible navigation.

Command languages

The original style of computer-human interaction is command language. The user types in requests through an artificial language.

Advantages and Disadvantages of command language dialog style.

Command language dialog styles have particular advantages and disadvantages to other dialog styles. These are included in the following table.

Advantages	**Disadvantages**
Powerful	Difficult to learn
Flexible, user controlled	Difficult to remember
Fast, efficient	Assumes typing skill
Uses minimal screen "real estate"	Error prone
	Enhancements are invisible

Command language is most appropriate for:
USER PSYCHOLOGY
Positive attitude
High motivation
KNOWLEDGE AND EXPERIENCE
Moderate to high typing skill
High system experience
High task experience
High application experience
infrequent use of other systems
High computer literacy
JOB AND TASK CHARACTERISTICS
High frequency of use
Formal training
Mandatory use
Low turnover rate
High task importance
Low task structure

A very powerful dialog style is created if care is taken in designing command languages. Because they are the oldest and were the most common used, command languages have been more extensively studied.

The design of command languages can be divided into four separate design issues:

- Command language semantics
- Command language syntax
- Command language lexicon
- Command language interaction

Command language semantics
Semantics describes the set of functionality the language provides and how that functionality is broken-down into language elements. Two extremes are rich language and minimal language. A study [Kraut, 1983] looked at the amount of richness to provide into a language. They stated that without reducing the richness of functionally the performance could be improved. This could be done by changing the design, such as more consistency in syntactic rules, easy access to status information, a more interactive style, better error feedback, and even alternative interfaces to more and less frequent used commands. Another study [Good, 1985] covered the same area.

Command language syntax
The syntax of a command language involves both the format and punctuation. From two studies [Ledgard *et al.*, 1980; Barnard & Grudin 1988] the following guidelines may be drawn:

- Provide consistency in syntax.
- Use an action object syntax. Avoid arbitrary use of punctuation.
- Avoid positioned grammars.
- The syntax should be natural and mnemonic.
- Defaulting of optional parameters.
- Avoid use of control keys.

Command language lexicon
The language lexicon in a command language is mainly command names and/or abbreviations. The studies from Bernard & Grudin have investigated the naming behaviour of both users and designers. There is a very low probability that any two users or designers will suggest the same name for a given function. The main problem here is to discover what properties of a command name set will provide optimal performance. In one study [Caroll, 1982] users performed better on congruent as opposed to non congruent commands. The best performance was reached if the commands are hierarchical and congruent. Several more studies [Furnas, 1985; Wixon, 1983]have been conducted and the following rules could be learned from all of them:

- Command names must be hierarchical, congruent, specific, familiar, consistent.
- Use user jargon and avoid computer jargon.
- Use consistent rule for abbreviations.
- Allow full command names.

Command language interaction

The structure of the dialog or interaction can be designed to enhance the usability of a command language system.
One study [Granda & Teitelbaum, 1982] investigated the location of the command line The head movement seemed to be an important time factor. The performance was optimal with the command line at the bottom of the screen. Most command languages start at the top and move down to the bottom. Several more guidelines may be used:

- Interactive support through defaults, commands editing, intelligent interpretation, type ahead, and feedback. On-line quick help.
- Function keys for high frequency commands.
- If possible use tailorable language.

Function keys

When using a function key interface, commands and sometimes objects are specified by pressing special keys on the keyboard.

Advantages and Disadvantages of function keys dialog style.

Function key language dialog styles have particular advantages and disadvantages to other dialog styles. These are included in the following table.

Advantages	**Disadvantages**
Self explanatory	Limited number of keys available
Easy to use	Hardware approaches to expansion are expensive.
Flexible	Software approaches to expansion sacrifice screen space efficiency and ease of use.
Requires little or no screen "real estate"	Makes keyboard system or application specific.
Low typing requirements	

Function keys are most appropriate for:
USER PSYCHOLOGY
Negative to positive attitude
Low to high motivation
KNOWLEDGE AND EXPERIENCE
Low typing skill
Low to high system experience
Moderate to high task experience
Moderate application experience
Low to high use of other systems
Moderate to high computer literacy
JOB AND TASK CHARACTERISTICS
Low to High frequency of use
Little or no training
Discretionary use
Moderate turnover rate
Moderate task importance
Low task structure
Limited task domain

Little or no research has been conducted that directly studies user performance with function key interfaces. The basic guidelines are drawn from other dialog styles:

- Provide enough function keys to support functionality, but not so many that scanning and finding them is difficult.
- Use function keys for generic, high frequency, important functions.
- Arrange in distinguished groups.
- Base groupings on flow of use.
- Recognizability takes precedence over consistency.
- Place high use keys within easy reach.
- Consistency in function keys.
- Provide feedback when function keys are pressed.
- Grey out soft function keys if not active and use status indicator on mode keys.

Direct manipulation

A direct manipulation interface is one in which users perform actions directly on visible objects. The term direct manipulation was first coined by Schniederman [Schneiderman, 1982] to describe new interfaces having the following characteristics:

- Continuous representation of objects
- Physical actions or labelled button presses in place of command language
- Rapid incremental reversible operations with immediately visible results

Direct manipulation interfaces often include pointing devices such as a mouse, trackball, or touch screen and often make heavy use of graphics in displaying objects and actions.

Advantages and Disadvantages of direct manipulation.

Direct manipulation dialog styles have particular advantages and disadvantages to other dialog styles. These are included in the following table.

Advantages	Disadvantages
Easy to learn and to remember	Not self-explanatory
What you see is what you get (WYSIWYG)	Can be inefficient
Flexible	Difficult to design recognisable icons
Provides context and instant, visual feedback	Icons take more screen "real estate" than words
Exploits human use of visual-spatial cues	
Less error prone	

Direct Manipulation is most appropriate for:
USER PSYCHOLOGY
Negative attitude
Low motivation
KNOWLEDGE AND EXPERIENCE
Low typing skill
Moderate system experience
Moderate to high task experience
Moderate application experience
High frequency of use of other systems
Low computer literacy
JOB AND TASK CHARACTERISTICS
Low frequency of use
Moderate training
Discretionary use
High turnover rate
Low task importance
Low task structure

Care must be taken in the design of a direct manipulation interface in order to realise the potential advantages. Several studies have tried to directly compare direct manipulation interfaces to non direct manipulation interfaces. One study [Robert & Moran, 1982] seems to suggest that direct manipulation interfaces are easier to use (for experts) and easier to learn (for novices) than non direct manipulation interfaces. Other studies showed a slight advantage of command language interface over the direct manipulation interface. Karat [Karat, 1987] found a clear advantage for the direct manipulation system when tested on novice users. Two other studies [Card & Moran, 1983; Shneiderman, 1982] showed clear efficiency advantages when using direct manipulation interfaces. All the studies more or less indicate whether or not direct manipulation will provide a performance advantage will probably depend, on how well the direct manipulation interface maps to users' goals, intentions, and tasks and what alternative interface it is being compared to. Icons must be carefully designed to allow natural and meaningful associations to be drawn between icon attributes and important object attributes.

Natural Language

Natural language allows users to express requests to a software application in there native language. For this propose a keyboard as an input device is assumed but it is also possible to provide voice input. Natural language interfaces are not yet widespread so only the advantages and disadvantages are given.

Advantages and Disadvantages of natural language.

Advantages	**Disadvantages**
Easy to learn	Assumes knowledge of the problem domain
Easy to remember	Lengthy confirmation and clarification dialogs
Less transfer problems to other languages	Assumes typing skill
Powerful	Error prone
Flexible	Enhancements are invisible
Fast	Expensive to implement
Uses moderate screen "real estate"	

Natural language is most appropriate for:
USER PSYCHOLOGY
Negative attitude
Low motivation
KNOWLEDGE AND EXPERIENCE
High typing skill
Low system experience
High task experience
Low application experience
High frequency of use of other systems
Low computer literacy
JOB AND TASK CHARACTERISTICS
Low frequency of use
Little or no training
Discretionary use
High turnover rate
Low task importance
Low task structure

D. Evaluation form

	Menu 1 2 3	Fill-in forms 1 2 3	Question and answers 1 2 3	Command language 1 2 3	Function keys 1 2 3	Direct manipulation 1 2 3	Natural language 1 2 3
User Psychology							
Attitude	Negativ●●●	Negativ●●●	Negativ●●●	Positiv●●●	Negati●●●	Negati●●●	Negativ●●●
Motivation	Low ●●●	Low ●●● Moderate	Low ●●●	High ●●●	Low ●●●	Low ●●●	Low ●●●
Knowledge and experience							
Typing skill	Low ●●●	Moderate ● High	Moderate ● High	Moderate ● High	Low ●●●	Low ●●●	High ●
System experience	Low ●●●	Low ●●● Moderate	Low ●● Moderate	High ●	Low ●●●	Low ●●●	Low ●●●
Task experience	Low ●●●	Modera●●● High	Low ●●●	High ●	Moder●●● High	Low ●●●	High ●●
Application experience	Low ●●●	Low ●●● Moderate	Modera●●●	High ●●●	Moder●●●	Low ●●●	Low ●●●
Use of other systems	frequen●●●	Modera●●● Frequent	Modera●●● Frequent	Infreq●●●	Infrequ●●●	Freque●●●	Frequen●●●
Computer literacy	Low ●●●	Moderate●● High	Low ●●●	High ●	Moderate ● High	Low ●●●	Low ●●●
Job and task characteristics							
Frequency of use	Low ●●●	Moderat●●● High	Low ●●●	High ● ●	Low ●●●	Low ●●●	Low ●●●
Primary training	Little o●●●	Some ●●●	Little o●●●	Forma●●●	Little o●●●	Little o●●●	Little o●●●
System use	Discreti●●●	Discreti●●●	Discreti●●●	Manda●●●	Discre●●●	Discre●●●	Discreti●●●
Turnover rate	High ●●●	Low ●●● Moderate	High ●●●	Low ●●●	Moder●●●	High ●●●	High ●●●
Other systems		Paper forms					
Task importance	Low ●●●	Moderat●●●	Low ●●●	High ●●	Moder●●●	Low ●●●	Low ●●●
Task structures	High ●●●	High ●●●	High ●●●	Low ●●	Low ●●● Moderate	Moder●●●	Low ●●
Operator M 1 ●	14	12	12	8	13	14	11
Operator A 2 ●	14	13	13	8	13	14	13
Maint/Process 3 ●	14	14	14	14	14	14	14

E. Touch Screen technologies

Force Vector

In force vector technology, a CRT or flat panel display is placed upon a platform. The platform consists of a top plate that rests upon three-dimensional springs moving in all directions relative to the base of the pedestal. Internal sensors measure the distance between the top plate and the base. Each sensor consists of two parallel plates that form a capacitor, one attached to the top plate and one attached to the base.

When the display is touched, the force causes a slight change in distance between the top plate and the base. This causes the distance between each set of sensor plates to change slightly, changing the capacitance across the plates and determining the amount the top plate moved in the x,y and z directions. The platform then determines the level of force exerted to cause this movement and the location where the force was applied

Strain Gage

The strain gage system consists of a CRT mounting shield, a clear glass overlay and four strain gage transducers hooked to the clear glass overlay. Touching the glass overlay causes the strain gage transducers at the four corners to measure, compare, and calculate the touch location.

Guided Acoustic Wave

Guided acoustic wave is based on transmitting acoustic waves through a glass overlay placed over the display surface. A transducer mounted on the edge of the glass emits an acoustic wave. The wave packet travels along the reflector array, is redirected across the overlay to the reflecting edge, and returns to the array where it is reflected back to the transducer. The first reflector will send a signal back first, then the second, and so on.

When a stylus such as a finger comes into contact with the wave, it attenuates the wave motion by absorbing part of the wave. Control electronics detect the location of the dip in the wave amplitude, thus determining the touch position.

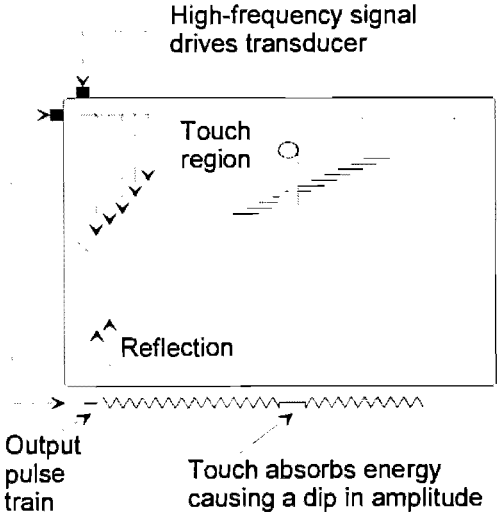


Figure 9, Guided Acoustic Wave

Surface Acoustic Wave

Surface acoustic wave technology is based on transmitting acoustic waves across the surface of a glass overlay placed over the displays surface. A touch screen controller sends a five-megahertz electrical signal to the transmitting transducer. The transducer mounted on the edge of the glass converts the signal into ultrasonic waves within the glass. These waves are directed across the front surface of the touch screen by an array of reflectors. Reflectors on the opposite side gather and direct the waves to the receiving transducer, which reconverts them into an electrical signal. Since the speed of the wave is known and the size of the glass overlay is fixed, the first reflector will send the first signal back first, then the second, and so on. When a stylus such as a finger comes into contact with the wave, it attenuates the wave motion by absorbing part of the wave. The received signal is compared to a stored signal, The change recognised and a coordinate calculated. This process happens independently for both X and Y axes. By measuring the amount of the signal that is absorbed, a Z-axis may also be determined.

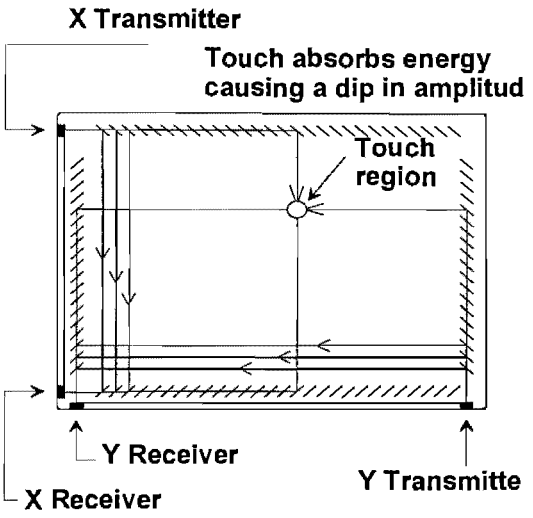


Figure 10, Surface Acoustic Wave

Capacitive

Capacitive overlay technology uses a glass overlay with a thin metallic coating over the surface of the display screen. Voltage is applied to the four corners of the screen creating a uniform voltage field. The user must touch the overlay with a conductive stylus, such as a finger, to activate the system. Touching the overlay surface causes a capacitive coupling with the voltage field, drawing a minute amount of current to the point of contact. The current flow from each corner is proportional to the distance to the finger and the ratios of these flows are measured by a controller and used to locate the touch.

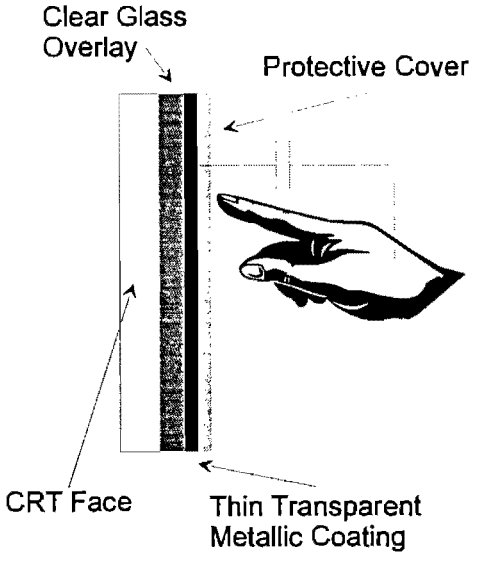


Figure 11, Capacitive

Infrared

Scanning infrared (IR) technology relies on the interruption of an IR light grid in front or the display screen. The touch frame or opto-matrix frame contains a row of IR-light emitting diodes (LED's) and photo transistors, each mounted on two opposite sides to create a grid of invisible infrared light. The frame assembly is comprised of printed wiring boards on which the optoelectronics are mounted and is concealed behind an IR-transparent bezel. The bezel shields the opto-electronics from the operating environment while allowing the IR beams to pass through.

The IR controller sequentially pulses the LED's to create a grid of IR light beams. When a stylus, such as a finger, enters the grid, it obstructs the beams. One or more phototransistors detect the absence of light and transmit a signal that identifies the x and y coordinates.

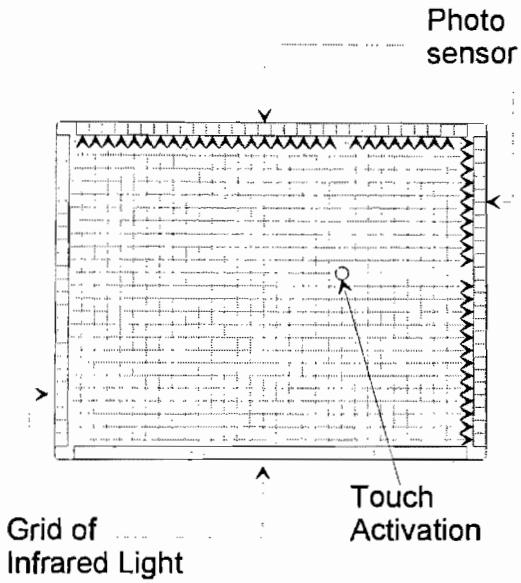


Figure 12, Infrared

Resistive

Resistive overlay technology consists of a glass overlay (substrate) with a thin metallic coating, over which a layer of polyester is placed (membrane). The polyester layer has a similar metallic coating on the interior surface. Tiny spacer dots of non-coated polyester prevent the two surfaces from contacting each other. A final hard coating is usually applied to the external surface of the polyester to reduce damage from sharp styli.

Digital, four- and five-wire designs are available for sensing the position of the touch. In the digital design , one layer is divided in to horizontal lines and the other layer is divided in to vertical lines. Together the two layers form a raster. When a finger or other stylus presses the two layers contact is made between a horizontal and a vertical line. The controller detects a specific block in the raster.

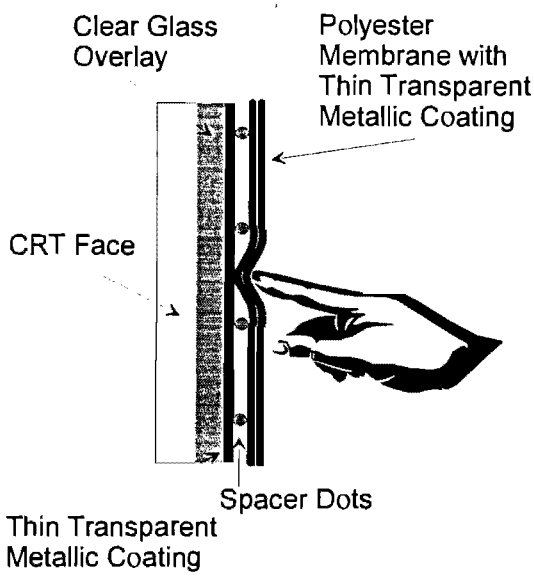


Figure 13, Resistive

In a four-wire-resistive touch screen, electrode arrays at opposite sides of the substrate can establish a 1-D voltage gradient across the substrate's resistive indium-tin-oxide (ITO) coating. Similar electrodes can establish an orthogonal gradient across the membrane's ITO coating. Both sets of electrodes also allow the ITO coatings to act as high-impedance probes. When the user touches a four-wire-system screen, the controller establishes a gradient across the substrate. The controller then measures the voltage at the point of touch using the membrane as a probe. Similarly, the controller establishes a gradient across the membrane and uses the substrate as a probe. The two voltages provide the x and y coordinates of the touch point.

In a five-wire system, the substrate has a resistive ITO coating and electrodes on all four sides. The membrane has a single electrode and a conductive coating. When a finger or other stylus presses the two layers together, the controller establishes first an x-axis and then a y-axis gradient across the substrate. The controller uses the membrane as a probe at all the times. The two voltages that the probe senses reflect the points x and y coordinates.

Touch System Comparison

Specific data regarding the impact of environmental factors on touch systems may be found in the following technology comparison charts.

Table 1 Touch System Comparison 1

	Stylus type	Resolution and Z-axis	Transparency	Sensor Drift and Calibration	Reliability	Activation, Parallax and Response Time
Force Vector	No stylus limitation	40 points/ inch plus 256 z-axis levels	100%	Subject to drift. Requires repetive calibration	80,000 hours MTBF	Tactile activation, no parallax, 200ms
Strain Gage	No stylus limitation	4096x4096 physical, plus z-axis	92%	Subject to drift. Requires complex repetive calibration	Not published by manufacture	Tactile activation, parallax, 100ms
Guided Acoustic Wave	Requires soft, energy-absorbing stylus	21,904 points/square inch, plus 256 z-axis levels	90% - 95%	Not subject to drift	Sensor Unlimited? Controller >180,000 hours MTBF	Tactile activation, no parallax, 18 - 50 ms
Surface Acoustic Wave	Requires soft, energy-absorbing stylus	100,000 points/square inch, plus 256 z-axis levels	90% - 95%	Not subject to drift	Sensor >50,000,000 touches per point Controller >65,000 hours MTBF	Tactile activation, no parallax, 18 - 22 ms
Capacitive	Requires conductive stylus	1024x1024 physical	85%	Not subject to drift**	Sensor 20,000,000 touches per point Controller >572,600 hours MTBF	Tactile activation 8-15 ms
Infrared	No stylus limitation Minimum stylus diameter 5/16 "	64 points/square inch no z-axis	100%	Not subject to drift	>138,000 hours MTBF	Proximity activation, parallax 18 - 40 ms
four-wire-resistive	No stylus limitation	100,000 points/square inch	55% - 75%	Subject to drift Requires repetive calibration	Sensor 2,000,000 touches per point Controller >180,000	Tactile activation 7 - 12 ms
five-wire-resistive	No stylus limitation	100,000 points/square inch	55% - 75%	Not subject to drift	Sensor 35,000,000 touches per point Controller >65,500 - >107,500 hours MTBF	Tactile activation

** Denied by the competition

Table 2 Touch System Comparison 2

	Scratch/Wear Resistance	NEMA Ratings, Moisture Resitance	Dust and Dirt Resistance	Chemical Resitance	Vibration and Shock Resistance
Force Vector	Extremely resistant	NEMA 12	Not affected by dust and dirt	Not effected by general purpose cleaning solutions	Sensitive to vibration and shock
Strain Gage	Extremely resistant	NEMA 12	Not affected by dust and dirt	Not effected by general purpose cleaning solutions	Sensitive to vibration and shock
Guided Acoustic Wave	Difficult to scratch, glass overlay is breakable	NEMA 12 NEMA 4	Not affected by dust and dirt	Not effected by general purpose cleaning solutions	
Surface Acoustic Wave	Difficult to scratch, glass overlay is breakable	NEMA 12	Will operate with moderate dust and dirt, excessive accumulation may affect performance	Not effected by general purpose cleaning solutions	Tolerant of vibration, glass overlay susceptible to shock
Capacitive	Difficult to scratch, conductive layer is subject to wear, glass overlay is breakable	NEMA 12 NEMA 4	Will operate with moderate dust and dirt, excessive accumulation may affect performance		Tolerant of vibration, thick glass overlay moderately susceptible to shock
Infrared	Extremely resistant	NEMA 12 NEMA 4	Will operate with moderate dust and dirt, excessive accumulation may affect performance	Not effected by general purpose cleaning solutions Chemicals that affect polycarbonates should not be used	Sensitive to vibration and shock
four-wire-resistive	Sensitive to damage of conductive coating on membrane	NEMA 12 NEMA 4	Not affected by dust and dirt	Not effected by general purpose cleaning solutions Chemicals that affect polyester should not be used.	Tolerant of vibration, glass overlay susceptible to shock
five-wire-resistive	Resistant (No failure before visible damage)	NEMA 12 NEMA 4	Not affected by dust and dirt	Not effected by general purpose cleaning solutions Chemicals that affect polyester should not be used.	Tolerant of vibration, glass overlay susceptible to shock

Manufacturer's published data

Explanation of used words

Drift - The gradual movement of the touch active zones away from the graphic targets representing them. Touch technologies that are subject to drift require periodic calibration to restore the touch active zones to the correct coordinates.

NEMA 4/12 complance - National Equipment Manufactuters Association. NEMA 4 compliance indicates that a device can withstand hose-directed water and still operate. NEMA 12 compliance indicates that a device is for industrial use.

Parallax - An optical phenomena in which a touch zone registers slightly differ from the graphical target. Parallax is caused by space between the display surface and the plane of the touch sensor, and varies according to the type and architecture of the touch technology. Parallax tends to be unnoticeable on flat displays and increases relative to the curvature of the display.

Resolution - The physical spacing between the adjacent touch coordinates.

Response time - The time required by the touch system to locate the touch and transmit the coordinates to the host system.

Stylus - A finger, digital pen, or any device used to activate the touch system.

Transparency - The clarity of the image, measured in percentage of light that is allowed to pass through the touch system. In overlay-based touch systems, clarity is reduced relative to the thickness of the overlay. Touch systems that do not use overlay-based technology have 100 % transparency.

Z-axis - The capability to measure pressure against the touch sensor, in additiond to x and y coordinates. Specific pressure thresholds can be set to emulate mouse button clicks and activate different touch events

MTBF - Mean Time Between Failure. A statistical estimate of how long a component or a system is expected to perform before a failure occurs

Comparison of the Touch Technologies

After giving an overview of the various touch technologies and the factors that can affect their suitability for use in various application environments, let us compare each technology's advantages and disadvantages.

Force Vector Advantages

- Resolution - Force vector touch systems provide a resolution of 40 Points per inch
- Z-axis - In addition to the typical x and y coordinates, force vector touch systems can provide a z-axis component, which is determined by the amount of pressure applied to the display.
- Transparency - Since no overlay covers the surface of the display, the transparency of the force vector technology is 100%
- Stylus Requirements - Force vector touch systems have no stylus limitations
- Integration - Since the display rests on the force vector touch pedestal, it does not require an invasive integration.

Force Vector Disadvantages

- Response time - Due to the complex calculations that are required by the force vector technology, the system's response time is relatively slow.
- Calibration - The calibration of a force vector touch system is complex and time consuming. Periodic adjustment will be required.
- Sealability - Force vector touch systems cannot be sealed to NEMA 4 requirements and are susceptible to direct exposure to water, dirt and corrosives.
- Environmental resistance - Force vector touch systems are sensitive to shock and vibration.

Strain Gage Advantages

- Resolution - Strain gage touch systems can deliver a resolution of up to 4096 x 4096 touch points.

- Z-axis - In addition to the typical x and y coordinates, stain gage technology can provide a z-axis component, which is determined by the amount of pressure applied to the display. The harder the user presses, the more energy is absorbed by the force-sensing transducers.
- Stylus Requirements - Strain gage touch systems have no stylus limitations.

Strain Gage Disadvantages

- Transparency - Strain gage touch systems typically have a transparency of 92%. This can affect image quality, particularly for high-resolution video mode.
- Calibration - The calibration of a strain gage touch system is complex and time consuming. Periodic adjustment may also be required.
- Sealability - Strain gage touch systems cannot be sealed to NEMA 4 requirements, and are susceptible to direct exposure to water, dirt and other corrosives.
- Parallax - Strain gage touch systems have some degree of parallax caused by the flat glass overlays combined with curved displays.

Guided Acoustic Wave Advantages

- Resolution - The resolution of guided acoustic wave technology is more than 100,000 touchpoints per square inch.
- Calibration - Since the reflector arrays determining the touch location are fixed in one place, guided acoustic wave systems are not subject to sensor drift. The touch system needs only to be aligned with its corresponding display.
- Z-axis - In addition to the typical x and y coordinates, guided acoustic wave systems can provide a z-axis component, which is determined by the amount of pressure applied to the sensor.
- Sealability - Guided acoustic wave touch systems sensors can be sealed to meet NEMA 4 requirements, preventing water and dirt from penetrating the display's internal electronics

Guided Acoustic Wave Disadvantage

- Transparency - Guided acoustic wave touch systems typically have a transparency between 90% and 95%. This can affect image quality, particularly for high resolution video mode.
- Stylus Requirements - A hard stylus, such as a pen, will not absorb the acoustic energy and will not be recognised as a touch.

Surface Acoustic Wave

- Resolution - The resolution of surface acoustic wave technology is determined by the physical placement of the reflector arrays. Typical resolution is 33 touch points per inch.
- Calibration - Since the reflector arrays determining the touch location are fixed in one place, surface acoustic wave systems are not subject to the phenomenon of sensor drift. The touch system needs only to be aligned with its corresponding display
- Z-axis - In addition to the typical x and y coordinates, surface acoustic wave systems can provide a z-axis component, which is determined by the amount of pressure applied to the sensor.
- Stylus requirements - Guided acoustic wave touch systems can be operated with gloved hand.
- Flexibility - Surface acoustic wave touch systems are adaptable to a wide range of displays without expensive custom fees.

Surface Acoustic Wave Disadvantages

- Transparency - Guided acoustic wave touch systems typically have a transparency between 90% and 95%. This can affect image quality, particularly for high resolution video mode.
- Stylus Requirements - A hard stylus, such as a pen, will not absorb the acoustic energy and will not be recognised as a touch.

Capacitive Advantages

- Resolution - Capacitive overlay systems can deliver a resolution of up to 4096 x 4096 touch points.
- Sealability - Capacitive overlay systems sensors can be sealed to the display, preventing water and dirt from penetrating the display's inner electronics. NEMA 4 requirements can be met.
- Flexibility - Surface acoustic wave touch systems are adaptable to a wide range of displays without expensive custom fees.

Capacitive Disadvantages

- Transparency - Capacitive touch systems typically have a transparency of 85%. This can affect image quality, particularly for high resolution video mode
- Stylus Requirements - Capacitive touch systems require a conductive stylus to operate. Materials that will not conduct a current, such as a pencil, fingernail, or insulated glove, will not activate the system. Although a capacitive system's sensitivity can be adjusted to activate with a thin cotton or surgically gloved hand, it cannot recognise a gloved and an ungloved hand using the same sensitivity setting.
- Sensor Drift/Calibration - Capacitive touch systems are subject to drift, where the touch-active zones move from the graphic targets representing them. Periodic calibration is required. The drift is denied by the manufacturer of capacitive touch systems.

Infrared Advantages

- Transparency - Because there is no overlay covering the display, the Transparency of infrared touch systems is 100%.
- Stylus Requirements - Scanning infrared touch systems require that the stylus have a minimum diameter of 5/16". However, there are no limitations on the type of material the stylus is made from.
- Calibration - Since the optoelectronics determining the touch location are fixed in one place, infrared touch systems are not subject to sensor drift. They need only to be aligned with the corresponding display.
- Sealability - Scanning infrared touch system can be sealed to NEMA 4 requirements, preventing water and dirt from penetrating the display electronics.

Infrared Disadvantages

- Resolution - Scanning infrared touch systems typically provide a resolution of eight touch points per inch.
- Parallax - Parallax occurs when a touch is detected while the stylus is still some small distance from the surface of the display. For scanning IR touch systems, parallax accurs because the invisible grid of IR beams can be interrupted before actual contact is made with the display. The amount of parallax is dependent on the type of integration and display type. Typically, the flatter the display face, the lower the amount of parallax.

Resistive Advantages

- Resolution - Resistive overlay touch systems can deliver resolution of up to 4069 x 4069 touch points.
- Stylus Requirements - Resistive overlay touch systems have no stylus limitations.
- Sealability - Resistive overlay touch systems can be sealed to NEMA 4 requirements, preventing water and dirt from penetrating the display's internal electronics.

Resistive Disadvantages

- Transparency - Resistive overlay touch systems have a transparency of 55% to 78% due to multiple layers of different materials found in the resistive overlay sensor. They are also more susceptible to glare and reflection than any other touch technology.
- Calibration - The 4-wire resistive overlay touch system are subject to drift. Periodic calibration is required. The 5-wire resistive overlay touch system is not subject to sensor drift.
- Environmental Resistance - The 4-wire resistive overlay touch system will degrade accuracy over time by constant flexing. The life time of the 4-wire is much shorter than the 5-wire technology. The exposed polyester top layer is susceptible to cuts, scratches, and abrasions.

Selection of touch technologies

Looking at the advantages and disadvantages of the various touch technologies and considering the application environment (not the EMC/ESD constrains) there are three technologies that suits the MMM equipment best: Surface Acoustic Wave, Capacitive and Five-wire-resistive.

The surface acoustic wave has a long life time (>50,000,000 touches per point), is difficult to destroy, has an image clarity that is good for an overlay and drift free operation is guaranteed. Like the surface acoustic wave technology the capacitive overlay is difficult to destroy and the life time is long (>20,000,000 touches per point). The Image clarity is less but still good for an overlay, and the response time is very fast (8-15 ms). Finally the five wire resistive with its ability to react on any kind of stylus, and its response time of 7-21 ms. The Transparency is low (55% - 75%). There is no sensor drift. All three technologies deliver very high resolutions and proved their durability in industrial environments. As may be noticed, the Infrared technology is banned because of the disadvantages in combination with CRT's. Although this technology is frequently used in industrial environment but only in combination with LCD and no constrains on ESD!!

F. circumstances use-case

Use case: (user)

Name:	Login
Prerequisites:	System is running. No user is logged in.
Description:	One of the users logged in with his name and password. The system verifies user data. The system displays startup screen. (some users may not like to be logged in)
Exceptions:	If user data is inconsistent with system user data an error message is given
Result:	One user is logged in
Name:	Measure Convergence
Prerequisites:	User is logged in. Colour recognition is done. Tube supplies are adjusted. Tube file is open.
Description:	
Exceptions:	
Result:	
Name:	Measure Eccentricity
Prerequisites:	
Description:	
Exceptions:	
Result:	
Name:	Measure Linewidth
Prerequisites:	
Description:	
Exceptions:	
Result:	
Name:	Select tube file name
Prerequisites:	User is logged in. One can select the right tube file or one can alter one to be used as a new one.
Description:	The system displays the directory with the available tube types and asks the user to select one or alter one. (1) The user selects a tube. The system connects the tube file to the tube. (2) The user selects a file to alter. Alters the contents of the file and saves it with a new tube name. The system connects the new tube file to the tube.
Exceptions:	If a new tube file has the same name as an old one an error message is displayed and the user is asked to enter a new tube file.
Result:	The parameters in the tube file are connected to the tube.
Name:	Edit program parameters
Prerequisites:	User is logged in and is authorised to edit parameters.
Description:	The user is able to change: ID number, Fields for system, Number of measurements, Meas. delay for 2A-mon., SBIP flag, Configuration, and Measurement. For more explanation see topics in manual
Exceptions:	
Result:	System has new program parameters
Name:	Edit tube parameters
Prerequisites:	User is logged in and is authorised to edit parameters
Description:	The user is able to change parameters which do not belong to a specific field: Tube name, Tube type, Delta , Delta gain, Video Sens. Factor X.,Video Sens. Factor Y., Delta Centre X. Delta Centre Y. Video Shift Step Size X., Nr. Of Video Shift Steps., Pulse Width, 1/2 Horz. Field Of View.,1/2 Vert Field Of View, Line frequency., Frame Frequency, Clock Pulse Type
Exceptions:	
Result:	System has new tube parameters which do not belong to a specific field
Name:	Edit position parameters
Prerequisites:	User is logged in and is authorised to edit parameters
Description:	The user is able to edit the mask pitch of a measure field
Exceptions:	If the measured field is not defined an error message is displayed..
Result:	The mask pitch is known by the system
Name:	Edit colour recognition settings
Prerequisites:	User is logged in and is authorised to edit parameters
Description:	The user is able to fill in the starting gain value for automatic gain adjustment. The user is also able to change the threshold for noise reduction.
Exceptions:	
Result:	Start gain for auto gain and threshold value are known by the system.
Name:	Edit convergence settings
Prerequisites:	User is logged in and is authorised to edit parameters
Description:	The user is able to edit the initial gain and initial threshold values used in the function search dot.
Exceptions:	
Result:	The initial gain value and the Initial threshold value are known by the system.
Name:	Edit Eccentricity settings
Prerequisites:	User is logged in and is authorised to edit parameters
Description:	The user is able to edit calibration parameters(gain(sense) and offset) and Camera parameters (gain and threshold) used for eccentricity measurements.
Exceptions:	
Result:	Sensitivity offset, gain, and threshold for eccentricity measurements are known by the system.
Name:	Edit linewidth settings
Prerequisites:	User is logged in and is authorised to edit parameters
Description:	The user is able to set the process parameters and the gain/threshold parameters for the line width process.

Exceptions:
Result: The parameters for the linewidth process are known by the system.

Name: Edit password
Prerequisites: User is logged in and is authorised to edit parameters
Description: The user can change his/hers password
Exceptions:
Result: User data contains new password.

Name: Save to disk
Prerequisites:
Description: The user can enable or disable the data savings to disk
Exceptions:
Result: Data to disk is enabled or disabled.

Name: Save to Aqua
Prerequisites:
Description: The user can enable or disable data transfer to AQUA
Exceptions:
Result: On line data collection is enabled or disabled.

Name: Use colour simultaneous utilities
Prerequisites:
Description: The user is able to perform a complete colour recognition. Before this function is performed the use must fill in the field of measurement and the expected white level. And the user or XY table must position the camera to the selected field. The system will then perform the following sequence.
Select <Red><Blanked> raster
Call SBIP function <Auto gain>
Call SBIP funktion <Mask to plane> for red
Select <Green> <Blanked> raster
Call SBIP function <Auto gain>
Call SBIP funktion <Mask to plane> for green
Select <Blue> <Blanked> raster
Call SBIP function <Auto gain>
Call SBIP function <Mask to plane> for blue
Exceptions:
Result: The system knows which dots are red, green, or blue.

Name: Use normalise video search utilities
Prerequisites:
Description: Video address is set by the system to the normal video position.
Exceptions:
Result: Video address is at the normal video position

Name: Use Video control utilities
Prerequisites:
Description: The user is able to select the colours, to set the video position and to choose a video picture. (more info in manual).
Exceptions:
Result: A screen colour is selected. A screen picture is selected and the line(s) or spot(s) are moved to specific positions.

Name: Use SBIP functions
Prerequisites:
Description: The user is able to:
reboot the SBIP
Initialise the SBIP
Put the SBIP to live video
Set Gain and threshold for specific fields manually
Process auto gain
Process auto threshold
Mask in Plane (make a one bit plane of a one coloured image)
Pixel Calibration
Show planes
Histogram
Exceptions:
Result:

Name: Use measurement utilities
Prerequisites:
Description: The user is able to execute some elementary measuring functions for analysing/development purposes:
Convergence
Gravity XY
Gravity X
GravityY
Brightness
Bright colour
Linewidth
Exceptions:
Result:

Use case: *(tube)*
Name: Project image
Prerequisites: Camera is in front of the field to measure.
Description: The system is able to project an image on the screen. The system is able to measure the image or to collect the image in a video memory for further analysis.
Exceptions:
Result: Image is collected in memory.

Use case: (remote system)
Sbip Functions

Name: BootSbip
Prerequisites:
Description:
Exceptions:
Result:

Name: InitTMS
Prerequisites:
Description:
Exceptions:
Result:

Name: GetImage
Prerequisites:
Description:
Exceptions:
Result:

Name: GrabImage
Prerequisites:
Description:
Exceptions:
Result:

Name: SetGain
Prerequisites:
Description:
Exceptions:
Result:

Name: AutoGain
Prerequisites:
Description:
Exceptions:
Result:

Name: AutoTreshold
Prerequisites:
Description:
Exceptions:
Result:

Name: Take Image toPlane
Prerequisites:
Description:
Exceptions:
Result:

Name: PixelCalibration
Prerequisites:
Description:
Exceptions:
Result:

Name: GravityX
Prerequisites:
Description:
Exceptions:
Result:

Name: GravityY
Prerequisites:
Description:
Exceptions:
Result:

Name: GravityXY
Prerequisites:
Description:
Exceptions:
Result:

Name: Border
Prerequisites:
Description:
Exceptions:
Result:

Name: Convergence
Prerequisites:
Description:
Exceptions:
Result:

Name: BrightnessNoColour
Prerequisites:
Description:
Exceptions:
Result:

Name: BrightnessColour
Prerequisites:
Description:
Exceptions:
Result:

Name: LineWidth
Prerequisites:

Description:
Exceptions:
Result:

Video/Sync functions

Name: Init Video
Prerequisites:
Description:
Exceptions:
Result:

Name: Setmode
Prerequisites:
Description:
Exceptions:
Result:

Name: SelectRaster
Prerequisites:
Description:
Exceptions:
Result:

Name: SetDosPosition
Prerequisites:
Description:
Exceptions:
Result:

Name: SearchDot
Prerequisites:
Description:
Exceptions:
Result:

Name: NormalVideoPos
Prerequisites:
Description:
Exceptions:
Result:

Name: InitPattern
Prerequisites:
Description:
Exceptions:
Result:

Measure functions
Name: Convergence
Prerequisites:
Description:
Exceptions:
Result:

Name: Eccentricity
Prerequisites:
Description:
Exceptions:
Result:

Name: LineWidth
Prerequisites:
Description:
Exceptions:
Result:

Name: ColourRecognition
Prerequisites:
Description:
Exceptions:
Result:

XY controller

Name: sent XY coordinates
Prerequisites:
Description:
Exceptions:
Result:

G. List of candidate classes

AQUA
camera
colour simultane
configuration
convergence
delay
disk
eccentricity
Fields
file
gain
Histogram
ID number recognition
image
line
linewidth
Mask pitch
Measure
normalise video
parameters
password
Plane
remote system
SBIP
screen image
screen picture
spot
systeem
threshold
tube
tube supplies
user
2A-mon
video control

identification classes brainstorm

convergence
date
eccentricity
linewidth
maintanence engeneer
measurement system
operator
proces engeneer
power
power supply tube
platform
vaccum
video memory
XY controller

H. Model dictionary

Video search:
Attribute:
operations:
search the video dot on the screen

Color recognition:
Attribute:
operations:
sequence to detect the color of the spots

Convergence:
Attribute:
operations:
measuring sequence to measure the convergence

Eccentricity:
Attribute:
operations:
measure the data collection system for quality assurance

Line width:
Attribute:
operations:
measuring sequence to measure linewidth

measuring equipment:
Attribute:
operations:
Measure Convergence, Measure Eccentricity, Measure Linewidth

SBIP:
Attribute:
operations:
frame grabber card grabs a frame from the *camera*
live video,gain,threshold,image map
Boot,initialise,Fixed video,auto gain,auto threshold,mask in plane,pixel calibration,show planes,histogram,colour simultane

camera:
Attribute: gain
operations:
an apparatus to grab an *image* from the *tube*

XY controller:
Attribute:
operations:
representation in the system of a XY controller to position a single camera.
x position, y position,ready
move

Tube:
Attribute:
operations:
representatation in the system of a real tube (CVT or TVT)
Tube name, Tube type, Delta threshold, Delta gain, Video sensitivity factor x, Video sensitivity factor y, Delta centre x, Delta centre y, Video shift step size x, number of video shift steps, pulse width, 1/2 Horz. field of view, 1/2 Vert. field of view, Line frequency, Frame Frequency, Clock pulse type, ID

Field:
Attribute:
operations:
one of the 25 fields on the display face

Picture:
Attribute:
operations:
representation of the picture on the tube

User:
Attribute:
operations:
representation in the system of a real person
function, password, preferences

remote system:
Attribute:
operations:
representation in the system of a remote system (two directions)

tube supplies:
Attribute:
operations:
representation of the hardware tube supplies to generate an *image* on a *tube*

I. Questionnaire

User Profile Checklist

Job and Task Characteristics

Job catagories:
- Operator ☐
- Production chief ☐
- Process specialist ☐
- Maintenance man ☐

Turnover rate:
- High ☐
- Moderate ☐
- Low ☐

Frequency of use:
- Low ☐
- Medium ☐
- High ☐

Sytem use:
- Mandatory ☐
- Discretionary ☐

Primary training:
- None ☐
- Manual only ☐
- Elective formal ☐
- Mandatory formal ☐

Task importance:
- High ☐
- Low ☐

Task structure:
- High ☐
- Moderate ☐
- Low ☐

Psychological Characteristics

Cognitive styles:
- Verbal ☐ Spatial ☐
- Analiytic ☐ Intuitive ☐

Attitude:
- Positive ☐
- Neutral ☐
- Negative ☐

Motivation:
- High ☐
- Moderate ☐
- Low ☐

Knowledge and Experience

Reading level:
- Non ☐
- Low ☐
- Medium ☐
- High ☐

Typing skill:
- Non ☐
- Low ☐
- Medium ☐
- High ☐

Education:
- Non ☐
- Low ☐
- Medium ☐
- High ☐

System experience:
- Expert ☐
- Moderate ☐
- Novice ☐

Task experience:
- Novice in field ☐
- Moderate ☐
- Expert in field ☐

Application experience:
- No similar systems ☐
- One similar system ☐
- Some similar system ☐

Native language:
- English ☐
- German ☐
- Frence ☐
- Spanish ☐
- Taiwanish ☐
- Chinish ☐

Use of other systems:
- Little or none ☐
- Frequent ☐

Computer literacy:
- Non ☐
- Low ☐
- Moderate ☐
- High ☐

J. Header file of Tube class

```
//## begin module.cm preserve=no
//      %X% %Q% %Z% %W%
//## end module.cm

//## begin module.cp preserve=no
//## end module.cp

//## Module: CTube; Pseudo Package specification
//## Subsystem: <Top Level>
//## Source file: E:\CTube.h


#ifndef CTube_h
#define CTube_h 1

//## begin module.additionalIncludes preserve=no
//## end module.additionalIncludes

//## begin module.includes preserve=yes
//## end module.includes

// CMeasuring_Equipment
#include "CMsrngEq.h"
// CField
#include "CField.h"

//## begin module.additionalDeclarations preserve=yes
//## end module.additionalDeclarations


//## Class: CTube
//## Category: <Top Level>
//## Subsystem: <Top Level>
//## Persistence: Transient
//## Cardinality/Multiplicity: n


class CTube
{
 //## begin CTube.initialDeclarations preserve=yes
 //## end CTube.initialDeclarations

 public:
  //## Constructors (generated)
   CTube();

   CTube(const CTube &right);

  //## Destructor (generated)
   ~CTube();

  //## Equality Operations (generated)
   int operator==(const CTube &right) const;

   int operator!=(const CTube &right) const;


  //## Other Operations (specified)
   //## Operation: m_Save%884685593
   void m_Save();

   //## Operation: m_Load%884685594
   void m_Load();

   //## Operation: m_GetSelectedField%884685595
   void m_GetSelectedField();

   //## Operation: m_GetNextSelectedField%884685596
   void m_GetNextSelectedField();

  //## Get and Set Operations for Associations (generated)

   //## Association: Has Tube%34846EB700FA
   //## Role: CTube::measuring equipment
   const CMeasuring_Equipment get_measuring_equipment()
     const;
   void set_measuring_equipment(const
     CMeasuring_Equipment value);

   //## Association: <unnamed>%34846F380000
   //## Role: CTube::<the_CField>
   const CField get_the_CField(const int index) const;
   void set_the_CField(const int index, const CField value);

  // Additional Public Declarations
   //## begin CTube.public preserve=yes
   //## end CTube.public


protected:
  // Additional Protected Declarations
   //## begin CTube.protected preserve=yes
   //## end CTube.protected


private:
 //## Get and Set Operations for Class Attributes (generated)

   //## Attribute: m_name
   // Under tube parameter - Tube name - This text is also
   // added to the Data collection (on disk) if enabled. To
   // edit this name go to submenu - Tube name - and key in
   // the name (e.g. 66" FS 32 kHz 50 Hz). Confirm your
     change
   // via Change edit data.
   const string get_m_name() const;
   void set_m_name(const string value);

   //## Attribute: m_MeasDelayFor2A
   // With the program parameter - Meas. delay for IIA-mon. -
   // the delay time of the first measurement (after switching
   // over to Video dot) is defined. This parameter is only
   // required to use Coper-T on the IIA-monitor because of
   // the Vk-Ik circuit behaviour in the IIA. For non IIA
   // applications use zero seconds.
   const int get_m_MeasDelayFor2A() const;
   void set_m_MeasDelayFor2A(const int value);

   //## Attribute: m_Type
   // Under the tube parameter - Tube type - TVT or CMT tube
   // types can be selected. This choice is important for the
   // Colour recognition algorithm
   // · CMT  has a hexagonal dot structure;
   // · TVT  has an Inline structure.
   const enum get_m_Type() const;
   void set_m_Type(const enum value);

   //## Attribute: m_DeltaThreshold
   // This value will be added to the value as found after the
   // function auto threshold. Normally this function will be
   // used after the auto gain function and before the
   // convergence measurement to optimise signal/noise ratio.
   // The default value for - Delta threshold - is 10. To
   // check if this is sufficient, visually observe the
   // picture area outside a Video dot. If pixels light up
   // outside the Video dot increase the - Delta threshold -
   // value to e.g. 15. Changing the setting can be done via
   // submenu - Delta threshold - by keying in the desired
   // value.
   const int get_m_DeltaThreshold() const;
   void set_m_DeltaThreshold(const int value);
```

```
//## Attribute: m_DeltaGain
// The Gain as determined under the function Auto gain can
// be changed via the tube parameter - Delta gain -. The
// default value for - Delta gain - is 5.
const int get_m_DeltaGain() const;
void set_m_DeltaGain(const int value);

//## Attribute: m_VideoSensFactorX
// By the tube parameter - Video sens. factor X - is meant
// the distance (in mm) the vertical video line is shifted
// per video step X on the screen. This parameter enables
// the program to position the vertical video line (or
// video dot) in front of the camera centre. The value of -
// Video sens. factor X - depends on the tube size and the
// horizontal deflection (scan amplitude and fre-quency).
// This parameter is determined by Auto calibration (as
// indicated by (c), see Tube parameters). As
// autocalibration value it will be calculated for the
// centre of the tube.
// The method is as follows:
//
// 1. First the Video dot is positioned in front of the
// camera and the nominal video data X and data Y addresses
// are stored.
// 2. The Video dot position is changed to the nominal data
// X + 1 and data Y+1. The centre of Gravity XY is
// measured.
// 3. The video dot position is changed to the nominal data
// X-1 and data Y-1. Again the centre of gravity XY is
// measured.
// 4. The - Video sens. factor X - and - Video sens. factor
// Y - are calculated as follows:
//
// Video sens. factor X =      (gravity for data X-1 - gravity
// for data X+1)/2
// Video sens. factor Y =      (gravity for data Y-1 - gravity
// for data Y+1)/2
const int get_m_VideoSensFactorX() const;
void set_m_VideoSensFactorX(const int value);

//## Attribute: m_VideoSensFactorY
// Under the tube parameter - Video sens. factor Y - is
// meant the dis-tance (in mm) the horizontal video line or
// video dot is shifted per video step y on the screen.
// This parameter enables the program to position the
// horizontal video line or video dot in front of the
// camera centre. The value of - Video sens. factor Y -
// depends of the tube size and the vertical deflection
// (scan amplitude and frequency). This parameter is
// determined by Auto calibra-tion (as indicated by (c),
// see Tube parame-ters). For the determi-nation method,
// see topic Video sens. factor X.
const int get_m_VideoSensFactorY() const;
void set_m_VideoSensFactorY(const int value);

//## Attribute: m_DeltaCentreX
// The tube parameter - Delta centre X - is the permissible
// searching tolerance for the vertical video line or video
// dot position to the centre of the camera. If this
// parameter is adjusted too critical, the video line or
// video dot cannot be positioned. An excessive - Delta
// centre X - value may cause part of the Video dot to
// extend outside the camera Field of view. The - Delta
// centre X - is determined by Auto calibration (as
// indicated by (c), see Tube parameters) using the formula:
//
// Delta centre X = Video sens. factor X * 3/4.
const int get_m_DeltaCentreX() const;
void set_m_DeltaCentreX(const int value);

//## Attribute: m_DeltaCentreY
// The tube parameter - Delta centre Y - is the permissible
// searching tolerance for the horizontal video line
// position to the centre of the camera. If this parameter
// is adjusted too critical, the video line cannot be
// positioned. An excessive - Delta centre y - value may
// cause part of the video dot to extend outside the camera
// Field of view. The Delta centre Y is determined by Auto
// calibration (as indicated by (c), see Tube parameters)
// using the formula:
//
// Delta centre Y = Video sens. factor Y * 3/4.
const int get_m_DeltaCentreY() const;
void set_m_DeltaCentreY(const int value);

//## Attribute: m_VideoShiftStepSizeX
// The tube parameter - Video shift step size X - is the
// step size between two video steps during Micro stepping.
// This parameter is determined by Auto calibration (as
// indicated by (c), see Tube parameters) based on the
// Number of video shift steps as follows:
//
// 1. The Video dot is positioned in front of the camera
// centre and the video data X address is stored.
// 2. The video position is set to the data X value - 1.
// The centre of gravity is measured.
// 3. The video position is stepwise changed with n = 5
// video shifts (so 5, 10, 15, ...) until the delta Gravity
// X is exceeding two times the phosphor pitch.
// 4. The - Video shift step size X - is calculated as
// follows:
//
// Video shift step size X =  total number of video shifts
// *          phosphor pitch
//                                  delta
// gravity X * 2              number of video shift
// steps
//
// This results in a - Video shift step size X - which is
// required to reach a Video dot position change of one
// phosphor pitch by the chosen Nr. of video shift steps.
const int get_m_VideoShiftStepSizeX() const;
void set_m_VideoShiftStepSizeX(const int value);

//## Attribute: m_NrOfVideoShiftSteps
// The tube parameter - Nr. of video shift steps - is the
// number of video steps, the spot is shifted during Micro
// stepping in X-direction. For convergence measurements
// this is also the number of measure-ments for averaging.
// This parameter is not defined by Auto calibration but
// set by the user. The value depends of the required
// Accu-racy (the higher the better) but be aware that the
// measuring time is influenced. The default value for the
// - Nr. of video steps - is 5. To change this parameter go
// to submenu - Nr. of video shift steps - and key in the
// desired value. To store see, topic Change edit data. To
// check if the Accu-racy is within limits see topic
// Accuracy.
const int get_m_NrOfVideoShiftSteps() const;
void set_m_NrOfVideoShiftSteps(const int value);

//## Attribute: m_PulseWidth
// The tube parameter - Pulse width - determines the width
// of the Video dot or vertical line in horizontal
// direction. The value of the - Pulse width - depends on
// the line deflection frequency, the type of tube and the
// Test array on which the tube is measured. The proper
// value is determined empirically by increasing or
// decreasing the value until the Video dot on the camera
// image comprises 4 to 5 phosphor dots/lines of one colour
// in the horizontal direction.
const int get_m_PulseWidth() const;
void set_m_PulseWidth(const int value);

//## Attribute: m_1/2HorzFieldOfView
// The tube parameter - 1/2 Horz. field of view - stands
// for half the Field of view of the camera (in mm) in the
// X-direction. For Coper-T this parameter is not fixed
// depending to the choice of the enlargement by the
// camera's :
// default:
//     For TVT      : 6000 (mm).
```

```
//     For CMT     : 3000 (mm).
// The above value is fixed as tube parameter for the Tube
// type TVT/CMT in question.
//
// This value can be determined with:
// · select blank green video
// · focus the camera
// · select live video
// · count the number of horizontal trios
// · calculate 1/2 horizontal field of view using the
// formula:
//        1/2 * num. of trios * maskpitch
const int get_m_1X2HorzFieldOfView() const;
void set_m_1X2HorzFieldOfView(const int value);

//## Attribute: m_1/2VertFieldOfView
// The tube parameter - 1/2 Vert. field of view - stands
// for half the field of the camera (in mm ) in the
// Y-direction. Because the CCD pixel length/height ratio
// is fixed, this parameter is in fixed proportion to 1/2
// Horz. field of view. By Auto cali-bra-tion the Vert.
// field of view is set to:
//
//    1/2 Horz. field of view * 3/4 :
// default:
//    For TVT      : 4500 (mm).
//    For CMT      : 2250 (mm).
const int get_m_1X2VertFieldOfView() const;
void set_m_1X2VertFieldOfView(const int value);

//## Attribute: m_LineFrequency
// Under the tube parameter - Line frequency - the
// appropriate line deflection frequency must be stored.
// Therefore check the actual line deflection frequency
// setting of the time base unit and go to - Line frequency
// - and press <enter>. Make your choice in the submenu and
// store according procedure Change edit data.
//
// NOTE:  In case the - Line frequency - is changed from 16
// kHz into 32/62 kHz or vice verse the jumper setting on
// the video gener-ator card must be changed also.
const int get_m_LineFrequency() const;
void set_m_LineFrequency(const int value);

//## Attribute: m_FrameFrequency
// Under the tube parameter - Frame frequency - the
// appropriate frame deflection frequency must be stored.
// Therefore check the actual frame deflection setting of
// the time base unit.
const int get_m_FrameFrequency() const;
void set_m_FrameFrequency(const int value);

//## Attribute: m_ClockPulseType
// With the tube parameter - Clock pulse type - a selection
// can be made between  - Mains locked - or - Crystal
// locked - video synchronising. For normal measurements
// always select Mains locked. The - Crystal locked - mode
// is used only for special investigations. To change the
// setting go to submenu Clock pulse type and press
// <enter>. Make the appropriate choice Mains
// locked/Crystal locked and store according procedure
// Change edit data.
const enum get_m_ClockPulseType() const;
void set_m_ClockPulseType(const enum value);

//## Attribute: m_DAF gun
const boolean get_m_DAF_gun() const;
void set_m_DAF_gun(const boolean value);

// Additional Private Declarations
//## begin CTube.private preserve=yes
//## end CTube.private


private: //## implementation
// Data Members for Class Attributes

//## begin CTube::m_name.attr preserve=no  private: string
  {U}
string m_name;
//## end CTube::m_name.attr

//## begin CTube::m_MeasDelayFor2A.attr preserve=no
  private: int {U}
int m_MeasDelayFor2A;
//## end CTube::m_MeasDelayFor2A.attr

//## begin CTube::m_Type.attr preserve=no  private: enum
  {U}
enum m_Type;
//## end CTube::m_Type.attr

//## begin CTube::m_DeltaThreshold.attr preserve=no
  private: int {U}
int m_DeltaThreshold;
//## end CTube::m_DeltaThreshold.attr

//## begin CTube::m_DeltaGain.attr preserve=no  private: int
  {U}
int m_DeltaGain;
//## end CTube::m_DeltaGain.attr

//## begin CTube::m_VideoSensFactorX.attr preserve=no
  private: int {U}
int m_VideoSensFactorX;
//## end CTube::m_VideoSensFactorX.attr

//## begin CTube::m_VideoSensFactorY.attr preserve=no
  private: int {U}
int m_VideoSensFactorY;
//## end CTube::m_VideoSensFactorY.attr

//## begin CTube::m_DeltaCentreX.attr preserve=no
  private: int {U}
int m_DeltaCentreX;
//## end CTube::m_DeltaCentreX.attr

//## begin CTube::m_DeltaCentreY.attr preserve=no
  private: int {U}
int m_DeltaCentreY;
//## end CTube::m_DeltaCentreY.attr

//## begin CTube::m_VideoShiftStepSizeX.attr preserve=no
  private: int {U}
int m_VideoShiftStepSizeX;
//## end CTube::m_VideoShiftStepSizeX.attr

//## begin CTube::m_NrOfVideoShiftSteps.attr preserve=no
  private: int {U}
int m_NrOfVideoShiftSteps;
//## end CTube::m_NrOfVideoShiftSteps.attr

//## begin CTube::m_PulseWidth.attr preserve=no  private:
  int {U}
int m_PulseWidth;
//## end CTube::m_PulseWidth.attr

//## begin CTube::m_1/2HorzFieldOfView.attr preserve=no
  private: int {U}
int m_1X2HorzFieldOfView;
//## end CTube::m_1/2HorzFieldOfView.attr

//## begin CTube::m_1/2VertFieldOfView.attr preserve=no
  private: int {U}
int m_1X2VertFieldOfView;
//## end CTube::m_1/2VertFieldOfView.attr

//## begin CTube::m_LineFrequency.attr preserve=no
  private: int {U}
int m_LineFrequency;
//## end CTube::m_LineFrequency.attr
```

```
    //## begin CTube::m_FrameFrequency.attr preserve=no
      private: int {U}
    int m_FrameFrequency;
    //## end CTube::m_FrameFrequency.attr

    //## begin CTube::m_ClockPulseType.attr preserve=no
      private: enum {U}
    enum m_ClockPulseType;
    //## end CTube::m_ClockPulseType.attr

    //## begin CTube::m_DAF gun.attr preserve=no  private:
      boolean {U}
    boolean m_DAF_gun;
    //## end CTube::m_DAF gun.attr

   // Data Members for Associations

    //## Association: Has Tube%34846EB700FA
    //## begin CTube::measuring equipment.role preserve=no
      public: CMeasuring_Equipment { -> UHGN}
    CMeasuring_Equipment measuring_equipment;
    //## end CTube::measuring equipment.role

    //## Association: <unnamed>%34846F380000
    //## begin CTube::<the_CField>.role preserve=no  public:
      CField {1 -> 25UHN}
    CField the_CField[25];
    //## end CTube::<the_CField>.role

   // Additional Implementation Declarations
    //## begin CTube.implementation preserve=yes
    //## end CTube.implementation

};

// Class CTube

//## Get and Set Operations for Class Attributes (inline)

inline const string CTube::get_m_name() const
{
 //## begin CTube::get_m_name%.get preserve=no
 return m_name;
 //## end CTube::get_m_name%.get
}

inline void CTube::set_m_name(const string value)
{
 //## begin CTube::set_m_name%.set preserve=no
 m_name = value;
 //## end CTube::set_m_name%.set
}

inline const int CTube::get_m_MeasDelayFor2A() const
{
 //## begin CTube::get_m_MeasDelayFor2A%.get preserve=no
 return m_MeasDelayFor2A;
 //## end CTube::get_m_MeasDelayFor2A%.get
}

inline void CTube::set_m_MeasDelayFor2A(const int value)
{
 //## begin CTube::set_m_MeasDelayFor2A%.set preserve=no
 m_MeasDelayFor2A = value;
 //## end CTube::set_m_MeasDelayFor2A%.set
}

inline const enum CTube::get_m_Type() const
{
 //## begin CTube::get_m_Type%.get preserve=no
 return m_Type;
 //## end CTube::get_m_Type%.get
}

inline void CTube::set_m_Type(const enum value)
{
 //## begin CTube::set_m_Type%.set preserve=no
 m_Type = value;
 //## end CTube::set_m_Type%.set
}

inline const int CTube::get_m_DeltaThreshold() const
{
 //## begin CTube::get_m_DeltaThreshold%.get preserve=no
 return m_DeltaThreshold;
 //## end CTube::get_m_DeltaThreshold%.get
}

inline void CTube::set_m_DeltaThreshold(const int value)
{
 //## begin CTube::set_m_DeltaThreshold%.set preserve=no
 m_DeltaThreshold = value;
 //## end CTube::set_m_DeltaThreshold%.set
}

inline const int CTube::get_m_DeltaGain() const
{
 //## begin CTube::get_m_DeltaGain%.get preserve=no
 return m_DeltaGain;
 //## end CTube::get_m_DeltaGain%.get
}

inline void CTube::set_m_DeltaGain(const int value)
{
 //## begin CTube::set_m_DeltaGain%.set preserve=no
 m_DeltaGain = value;
 //## end CTube::set_m_DeltaGain%.set
}

inline const int CTube::get_m_VideoSensFactorX() const
{
 //## begin CTube::get_m_VideoSensFactorX%.get
      preserve=no
 return m_VideoSensFactorX;
 //## end CTube::get_m_VideoSensFactorX%.get
}

inline void CTube::set_m_VideoSensFactorX(const int value)
{
 //## begin CTube::set_m_VideoSensFactorX%.set preserve=no
 m_VideoSensFactorX = value;
 //## end CTube::set_m_VideoSensFactorX%.set
}

inline const int CTube::get_m_VideoSensFactorY() const
{
 //## begin CTube::get_m_VideoSensFactorY%.get
      preserve=no
 return m_VideoSensFactorY;
 //## end CTube::get_m_VideoSensFactorY%.get
}

inline void CTube::set_m_VideoSensFactorY(const int value)
{
 //## begin CTube::set_m_VideoSensFactorY%.set preserve=no
 m_VideoSensFactorY = value;
 //## end CTube::set_m_VideoSensFactorY%.set
}

inline const int CTube::get_m_DeltaCentreX() const
{
 //## begin CTube::get_m_DeltaCentreX%.get preserve=no
 return m_DeltaCentreX;
 //## end CTube::get_m_DeltaCentreX%.get
}

inline void CTube::set_m_DeltaCentreX(const int value)
{
 //## begin CTube::set_m_DeltaCentreX%.set preserve=no
 m_DeltaCentreX = value;
 //## end CTube::set_m_DeltaCentreX%.set
}
```

```
inline const int CTube::get_m_DeltaCentreY() const
{
 //## begin CTube::get_m_DeltaCentreY%.get preserve=no
 return m_DeltaCentreY;
 //## end CTube::get_m_DeltaCentreY%.get
}

inline void CTube::set_m_DeltaCentreY(const int value)
{
 //## begin CTube::set_m_DeltaCentreY%.set preserve=no
 m_DeltaCentreY = value;
 //## end CTube::set_m_DeltaCentreY%.set
}

inline const int CTube::get_m_VideoShiftStepSizeX() const
{
 //## begin CTube::get_m_VideoShiftStepSizeX%.get
      preserve=no
 return m_VideoShiftStepSizeX;
 //## end CTube::get_m_VideoShiftStepSizeX%.get
}

inline void CTube::set_m_VideoShiftStepSizeX(const int value)
{
 //## begin CTube::set_m_VideoShiftStepSizeX%.set
      preserve=no
 m_VideoShiftStepSizeX = value;
 //## end CTube::set_m_VideoShiftStepSizeX%.set
}

inline const int CTube::get_m_NrOfVideoShiftSteps() const
{
 //## begin CTube::get_m_NrOfVideoShiftSteps%.get
      preserve=no
 return m_NrOfVideoShiftSteps;
 //## end CTube::get_m_NrOfVideoShiftSteps%.get
}

inline void CTube::set_m_NrOfVideoShiftSteps(const int value)
{
 //## begin CTube::set_m_NrOfVideoShiftSteps%.set
      preserve=no
 m_NrOfVideoShiftSteps = value;
 //## end CTube::set_m_NrOfVideoShiftSteps%.set
}

inline const int CTube::get_m_PulseWidth() const
{
 //## begin CTube::get_m_PulseWidth%.get preserve=no
 return m_PulseWidth;
 //## end CTube::get_m_PulseWidth%.get
}

inline void CTube::set_m_PulseWidth(const int value)
{
 //## begin CTube::set_m_PulseWidth%.set preserve=no
 m_PulseWidth = value;
 //## end CTube::set_m_PulseWidth%.set
}

inline const int CTube::get_m_1X2HorzFieldOfView() const
{
 //## begin CTube::get_m_1X2HorzFieldOfView%.get
      preserve=no
 return m_1X2HorzFieldOfView;
 //## end CTube::get_m_1X2HorzFieldOfView%.get
}

inline void CTube::set_m_1X2HorzFieldOfView(const int
      value)
{
 //## begin CTube::set_m_1X2HorzFieldOfView%.set
      preserve=no
 m_1X2HorzFieldOfView = value;
 //## end CTube::set_m_1X2HorzFieldOfView%.set
}

inline const int CTube::get_m_1X2VertFieldOfView() const
{
 //## begin CTube::get_m_1X2VertFieldOfView%.get
      preserve=no
 return m_1X2VertFieldOfView;
 //## end CTube::get_m_1X2VertFieldOfView%.get
}

inline void CTube::set_m_1X2VertFieldOfView(const int value)
{
 //## begin CTube::set_m_1X2VertFieldOfView%.set
      preserve=no
 m_1X2VertFieldOfView = value;
 //## end CTube::set_m_1X2VertFieldOfView%.set
}

inline const int CTube::get_m_LineFrequency() const
{
 //## begin CTube::get_m_LineFrequency%.get preserve=no
 return m_LineFrequency;
 //## end CTube::get_m_LineFrequency%.get
}

inline void CTube::set_m_LineFrequency(const int value)
{
 //## begin CTube::set_m_LineFrequency%.set preserve=no
 m_LineFrequency = value;
 //## end CTube::set_m_LineFrequency%.set
}

inline const int CTube::get_m_FrameFrequency() const
{
 //## begin CTube::get_m_FrameFrequency%.get preserve=no
 return m_FrameFrequency;
 //## end CTube::get_m_FrameFrequency%.get
}

inline void CTube::set_m_FrameFrequency(const int value)
{
 //## begin CTube::set_m_FrameFrequency%.set preserve=no
 m_FrameFrequency = value;
 //## end CTube::set_m_FrameFrequency%.set
}

inline const enum CTube::get_m_ClockPulseType() const
{
 //## begin CTube::get_m_ClockPulseType%.get preserve=no
 return m_ClockPulseType;
 //## end CTube::get_m_ClockPulseType%.get
}

inline void CTube::set_m_ClockPulseType(const enum value)
{
 //## begin CTube::set_m_ClockPulseType%.set preserve=no
 m_ClockPulseType = value;
 //## end CTube::set_m_ClockPulseType%.set
}

inline const boolean CTube::get_m_DAF_gun() const
{
 //## begin CTube::get_m_DAF_gun%.get preserve=no
 return m_DAF_gun;
 //## end CTube::get_m_DAF_gun%.get
}

inline void CTube::set_m_DAF_gun(const boolean value)
{
 //## begin CTube::set_m_DAF_gun%.set preserve=no
 m_DAF_gun = value;
 //## end CTube::set_m_DAF_gun%.set
}

//## Get and Set Operations for Associations (inline)
```

```
inline const CMeasuring_Equipment
      CTube::get_measuring_equipment() const
{
 //## begin CTube::get_measuring_equipment%.get
      preserve=no
 return measuring_equipment;
 //## end CTube::get_measuring_equipment%.get
}

inline void CTube::set_measuring_equipment(const
      CMeasuring_Equipment value)
{
 //## begin CTube::set_measuring_equipment%.set preserve=no
 measuring_equipment = value;
 //## end CTube::set_measuring_equipment%.set
}

inline const CField CTube::get_the_CField(const int index)
      const
{
 //## begin CTube::get_the_CField%.get preserve=no
 return the_CField[index];
 //## end CTube::get_the_CField%.get
}

inline void CTube::set_the_CField(const int index, const CField
      value)
{
 //## begin CTube::set_the_CField%.set preserve=no
 the_CField[index] = value;
 //## end CTube::set_the_CField%.set
}


#endif
```

K. C code file of tube class

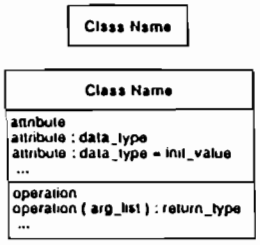
```
//## begin module.cm preserve=no
□
//        %X% %Q% %Z% %W%
□
//## end module.cm
□

□
//## begin module.cp preserve=no
□
//## end module.cp

//## Module: CTube; Pseudo Package body
//## Subsystem: <Top Level>
//## Source file: E:\CTube.cpp


//## begin module.additionalIncludes
preserve=no
//## end module.additionalIncludes

//## begin module.includes preserve=yes
//## end module.includes

// CTube
#include "CTube.h"

//## begin module.additionalDeclarations
preserve=yes
//## end module.additionalDeclarations


// Class CTube



CTube::CTube()
  //## begin CTube::CTube%.hasinit
preserve=no
  //## end CTube::CTube%.hasinit
  //## begin CTube::CTube%.initialization
preserve=yes
  //## end CTube::CTube%.initialization
{
  //## begin CTube::CTube%.body
preserve=yes
  //## end CTube::CTube%.body
}

CTube::CTube(const CTube &right)
  //## begin CTube::CTube%copy.hasinit
preserve=no
  //## end CTube::CTube%copy.hasinit
  //## begin
CTube::CTube%copy.initialization
preserve=yes
  //## end CTube::CTube%copy.initialization
{
  //## begin CTube::CTube%copy.body
preserve=yes
  //## end CTube::CTube%copy.body
}


CTube::~CTube()
{
  //## begin CTube::~CTube%.body
preserve=yes
  //## end CTube::~CTube%.body
}


int CTube::operator==(const CTube &right)
const
{
  //## begin CTube::operator==%.body
preserve=yes
  //## end CTube::operator==%.body
}

int CTube::operator!=(const CTube &right)
const
{
  //## begin CTube::operator!=%.body
preserve=yes
  //## end CTube::operator!=%.body
}



//## Other Operations (implementation)
void CTube::m_Save()
{
  //## begin CTube::m_Save%884685593.body
preserve=yes
  //## end CTube::m_Save%884685593.body
}

void CTube::m_Load()
{
  //## begin CTube::m_Load%884685594.body
preserve=yes
  //## end CTube::m_Load%884685594.body
}

void CTube::m_GetSelectedField()
{
  //## begin
CTube::m_GetSelectedField%884685595.body
preserve=yes
  //## end
CTube::m_GetSelectedField%884685595.body
}

void CTube::m_GetNextSelectedField()
{
  //## begin
CTube::m_GetNextSelectedField%884685596.bod
y preserve=yes
  //## end
CTube::m_GetNextSelectedField%884685596.bod
y
}


// Additional Declarations
//## begin CTube.declarations preserve=yes
//## end CTube.declarations
```

L. Touch screen producers

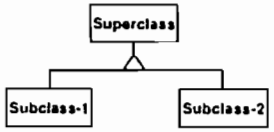
Company	Address1	City	State	PostalCode	Country
Carroll Touch	P.O. Box 1309	Round Rock	TX	78680	US
DEC/Data Concepts					US
Edmark					
Elo Touch Systems	41752 Christy St.	Fremont	CA	94538-3114	UK
Micro Touch	Unit 163 Milton Park	Didcot, Abingdon, Oxon		OX14 4SD	UK
Pixel Labs West					
VIVID Technology Corp.	1120 Sycamore Avenue, Suite 2a	Vista	CA	92083	US
Touch control	Building 101, Avenida de la Plata	Oceanside	CA	92056	US
CONTEC Microelectronics	2188 Bering Drive	San Jose	CA	95131	USA
Troll Touch	25510 Avenue Stanford, Ste. 106	Valencia	CA	91355	US
CyberTouch	853 Lawrence Drive	Nawbury Park	CA	91320-2232	US
Touch Technology		Austin	TX		US

Object Model Notation
Basic Concepts

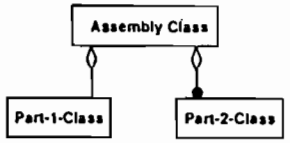
Class:



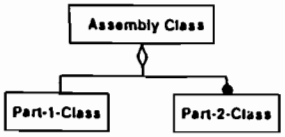
Generalization (inheritance):



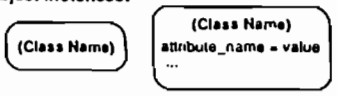
Aggregation:



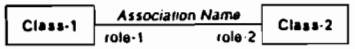
Aggregation (alternate form):



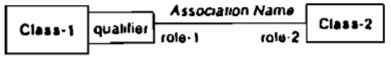
Object instances:



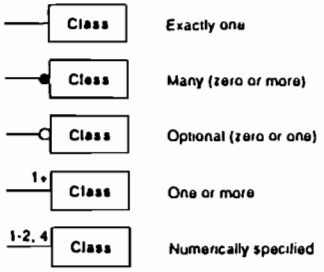
Association:



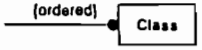
Qualified Association:



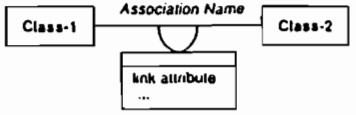
Multiplicity of Associations:



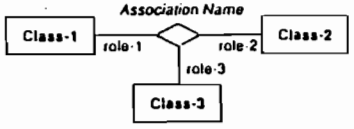
Ordering:



Link Attribute:



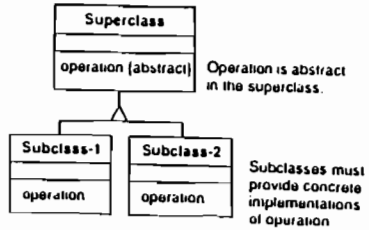
Ternary Association:



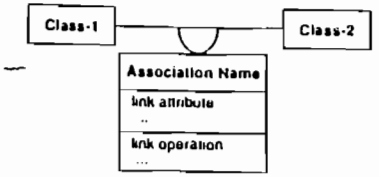
Instantiation Relationship:

Object Model Notation
Advanced Concepts

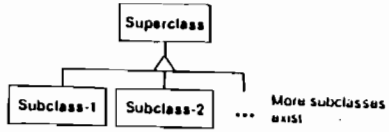
Abstract Operation:

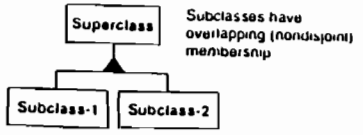


Association as Class:



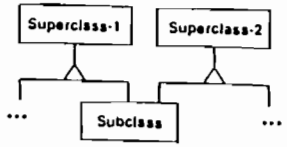
Generalization Properties:

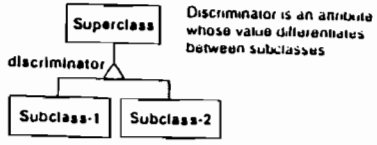




Subclasses have overlapping (nondisjoint) membership

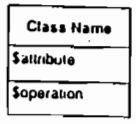
Multiple Inheritance:



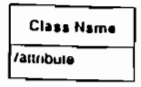


Discriminator is an attribute whose value differentiates between subclasses

Class Attributes and Class Operations:



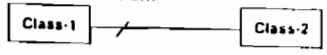
Derived Attribute:



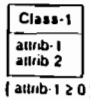
Derived Class:

Propagation of Operations:

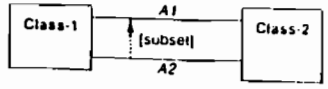
Derived Association:



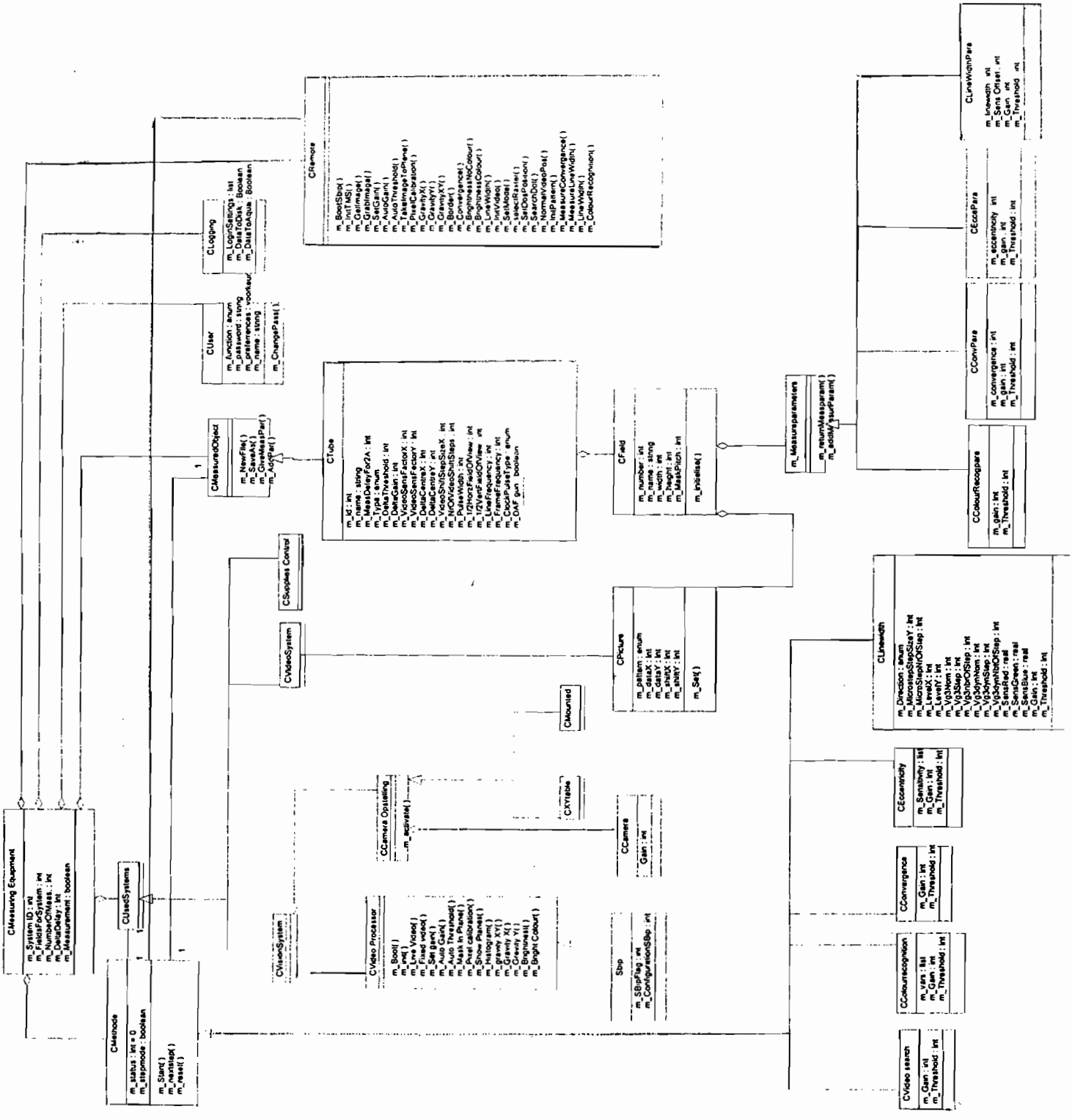
Constraints on Objects:



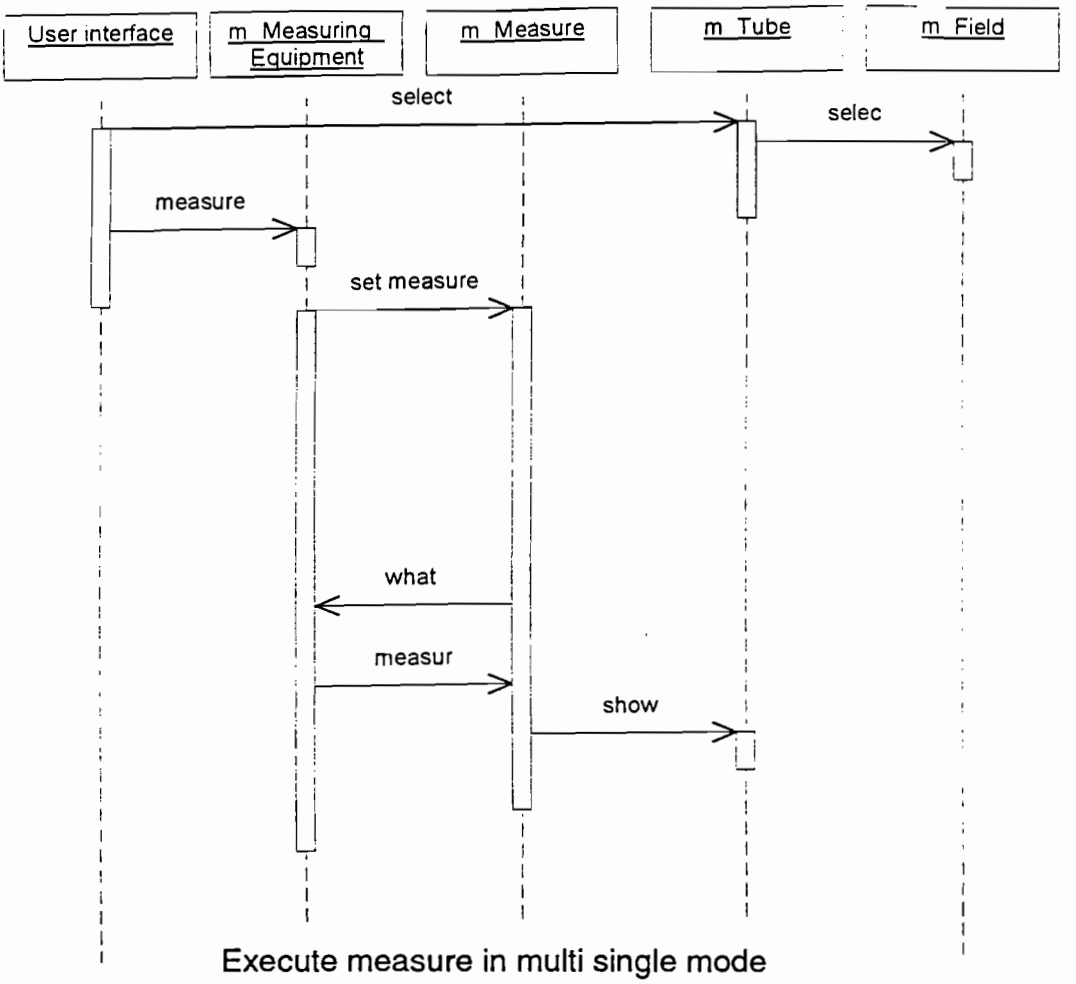
Constraint between Associations:



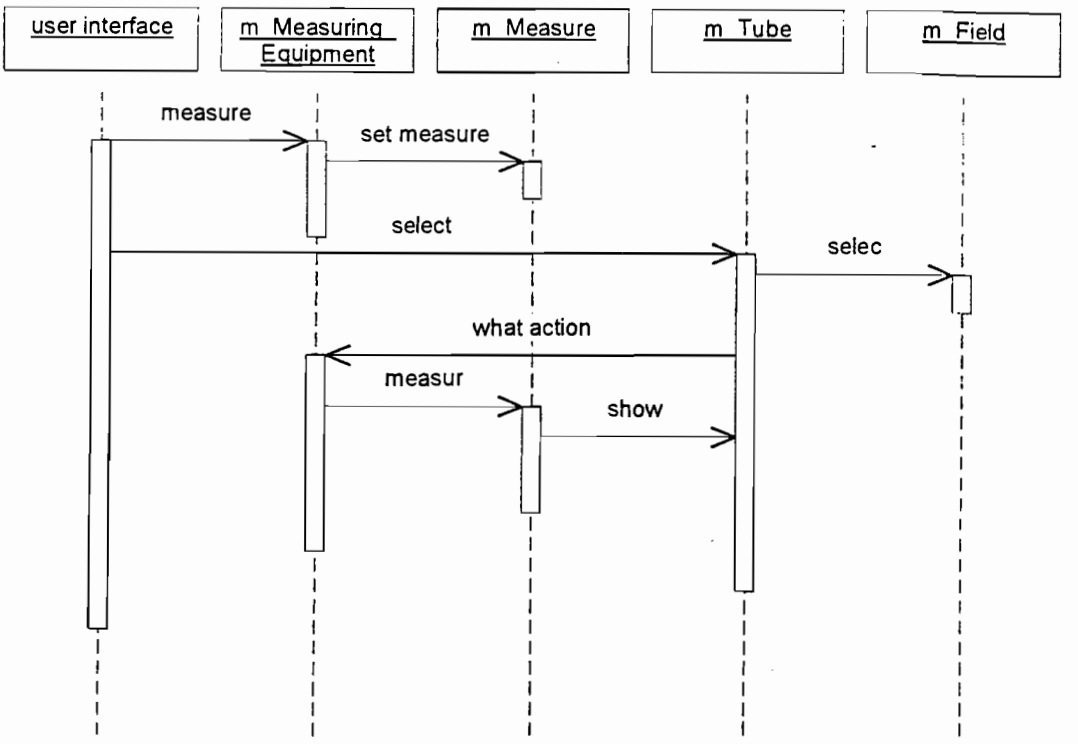
N. Design


CMeasuring Equipment
m_SystemID : int
m_FieldsForSystem : int
m_NumberOfMeas. : int
m_DelayDelay : int
m_Measurement : boolean
CUsedSystems
CMethode
m_status : int = 0
m_stepmode : boolean
m_Start()
m_nextstep()
m_reset()
CVisionSystem
CVideo Processor
m_Boot()
m_init()
m_Grab_Video()
m_Free_video()
m_Set gain()
m_Auto Gain()
m_Auto Threshold()
m_Mask In Plane()
m_Pixel calibration()
m_Show Planes()
m_Histogram()
m_gravity XY()
m_Gravity X()
m_Gravity Y()
m_Brightness()
m_Bright Colour()
CCamera Opsteling
m_activate()
CCamera
Gain : int
CXYTable
Ship
m_SBoxFlag : int
m_ConfigurationSBip : int
CVideoSystem
CSupplies Control
CMounted
CPicture
m_pattern : enum
m_deltaX : int
m_deltaY : int
m_shiftX : int
m_shiftY : int
m_Set()
CTube
m_Id : int
m_name : string
m_MeasDelayFor2A : int
m_Type : enum
m_DeltaThreshold : int
m_DeltaGain : int
m_VideoSkipFactorX : int
m_VideoSkipFactorY : int
m_DeltaCenterX : int
m_DeltaCenterY : int
m_VideoShiftStepSizeX : int
m_NrOfVideoShiftSteps : int
m_PulseWidth : int
m_1/2HorzFieldOfView : int
m_1/2VertFieldOfView : int
m_LineFrequency : int
m_FrameFrequency : int
m_OpAPulseType : enum
m_OAFgain : boolean
CField
m_number : int
m_name : string
m_width : int
m_height : int
m_MaskPitch : int
m_Initelisk()
m_Measureparameters
m_returnMeasparam()
m_addMeasureParam()
CMeasuredObject
m_NewFile()
m_SaveAs()
m_GiveMeasPar()
m_Activate()
CUser
m_function : enum
m_password : string
m_preferences : workset
m_name : string
m_ChangePass()
CLogging
m_LogInSettings : list
m_DataToDisk : Boolean
m_DataToAqua : Boolean
CRemote
m_BootStrap()
m_InitTMS()
m_GetImage()
m_GrabImage()
m_SetGain()
m_AutoGain()
m_AutoThreshold()
m_TakeImageToPlane()
m_PixelCalibration()
m_Gravity()
m_Gravity()
m_GravityXY()
m_Border()
m_Convergence()
m_BrightnessNoColour()
m_BrightnessColour()
m_LineWidth()
m_InitVideo()
m_SaveVideo()
m_LoadVideo()
m_SetDotPosition()
m_SearchDot()
m_NormalVideoPos()
m_InitPattern()
m_MeasureConvergence()
m_MeasureLineWidth()
m_LineWidth()
m_ColourRecognition()
CLineWidthPara
m_linewidth : int
m_Sens Offset : int
m_Gain : int
m_Threshold : int
CEccaPara
m_eccentricity : int
m_gain : int
m_Threshold : int
CConvPara
m_convergence : int
m_gain : int
m_Threshold : int
CColourRecogpara
m_gain : int
m_Threshold : int
CLineWidth
m_Direction : enum
m_MicrostepStepSizeY : int
m_MicrostepNrOfStep : int
m_LevelX : int
m_LevelY : int
m_Vg3Nom : int
m_Vg3Max : int
m_Vg3NrOfStep : int
m_Vg3AynNom : int
m_Vg3AynStep : int
m_Vg3AynNbOfStep : int
m_SensRed : real
m_SensGreen : real
m_SensBlue : real
m_Gain : int
m_Threshold : int
CEccentricity
m_Sensitivity : list
m_Gain : int
m_Threshold : int
CConvergence
m_Gain : int
m_Threshold : int
CColourrecognition
m_vars : list
m_Gain : int
m_Threshold : int
CVideo search
m_Gain : int
m_Threshold : int

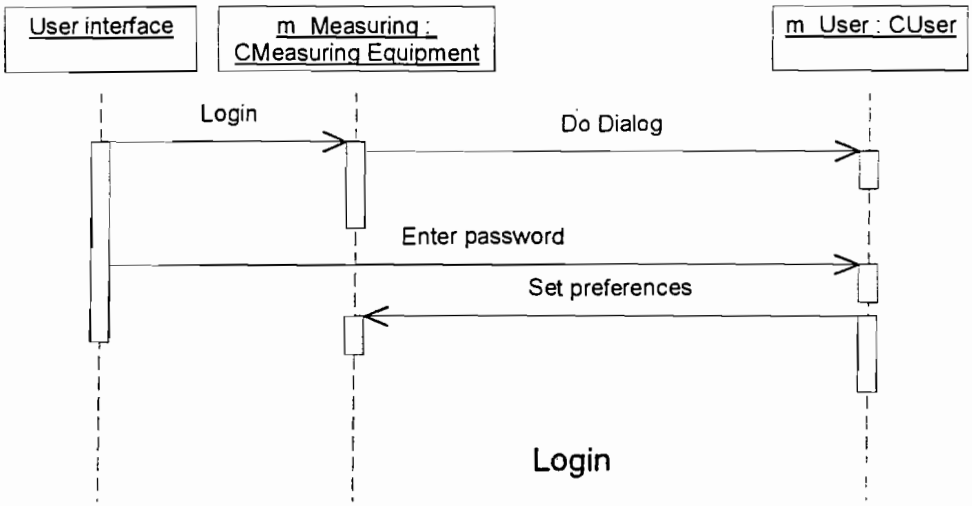
O. Internal scenarios



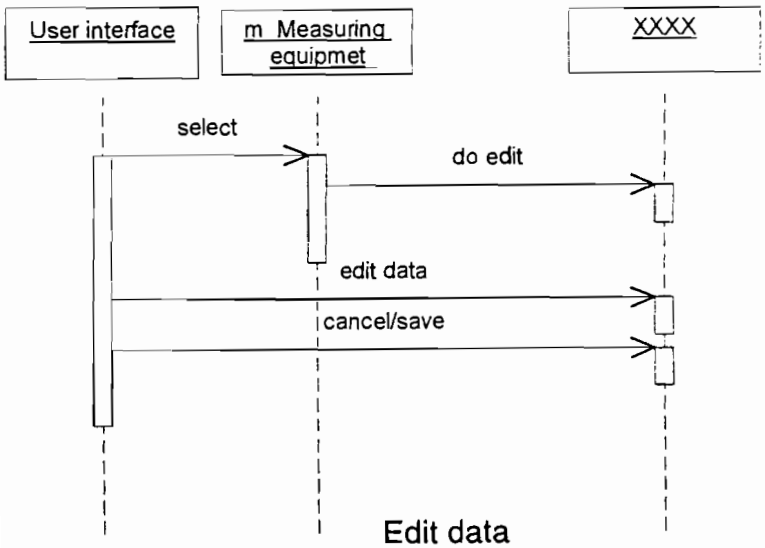
Execute measure in multi single mode



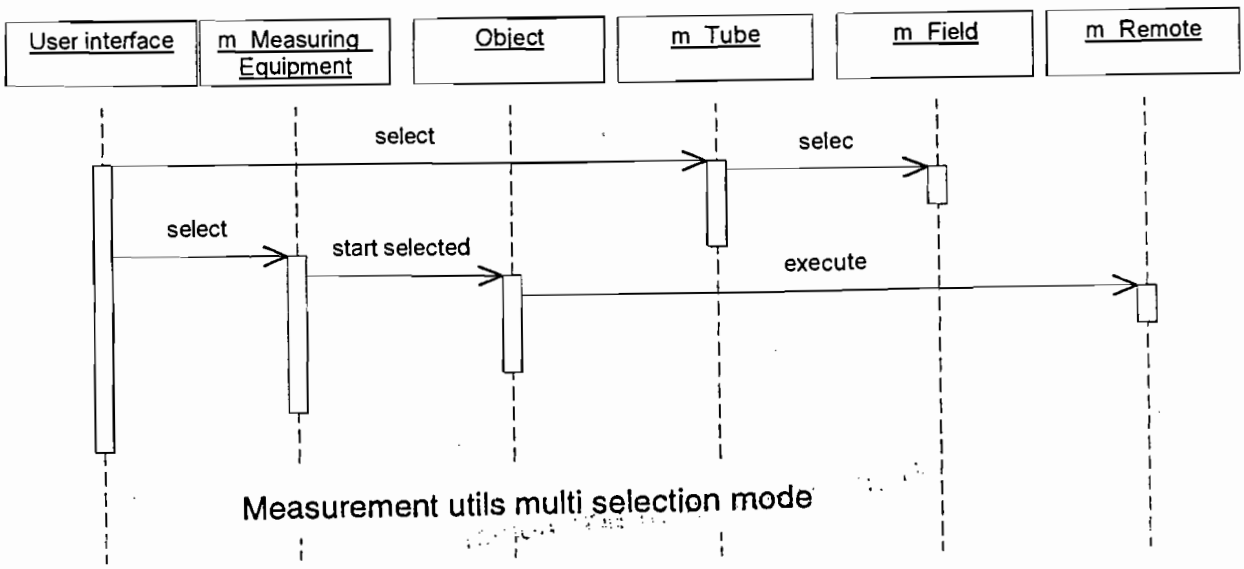
Execute measure in multi selection mode



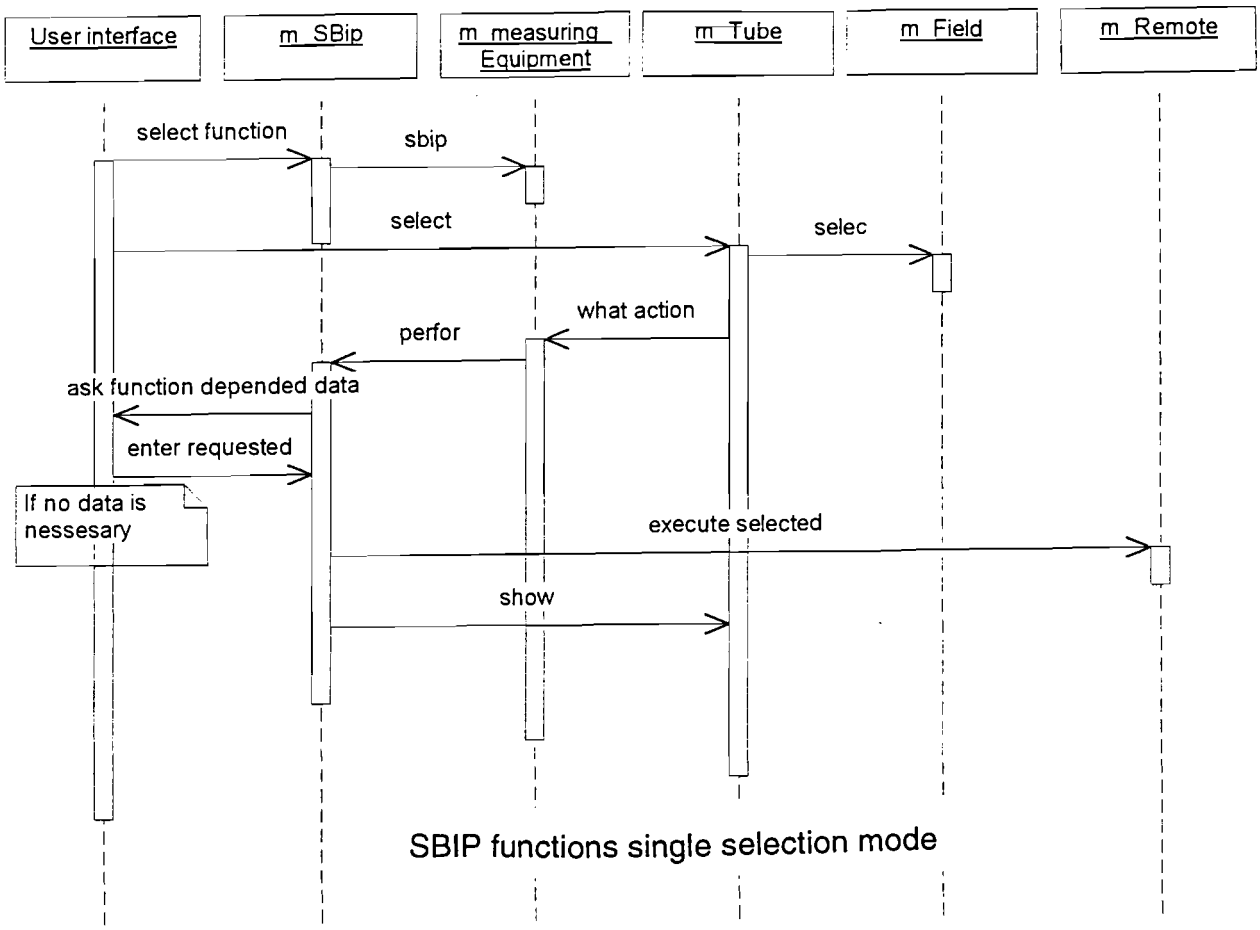
Login



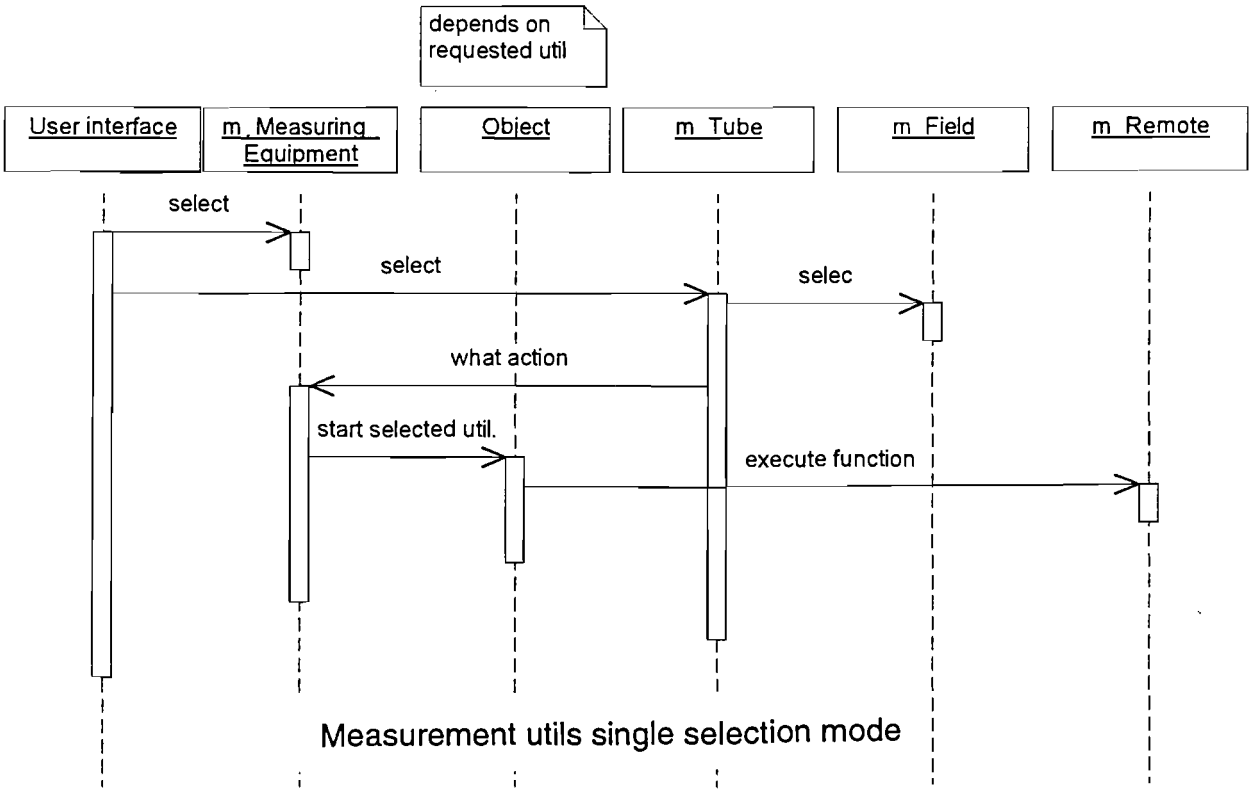
Edit data



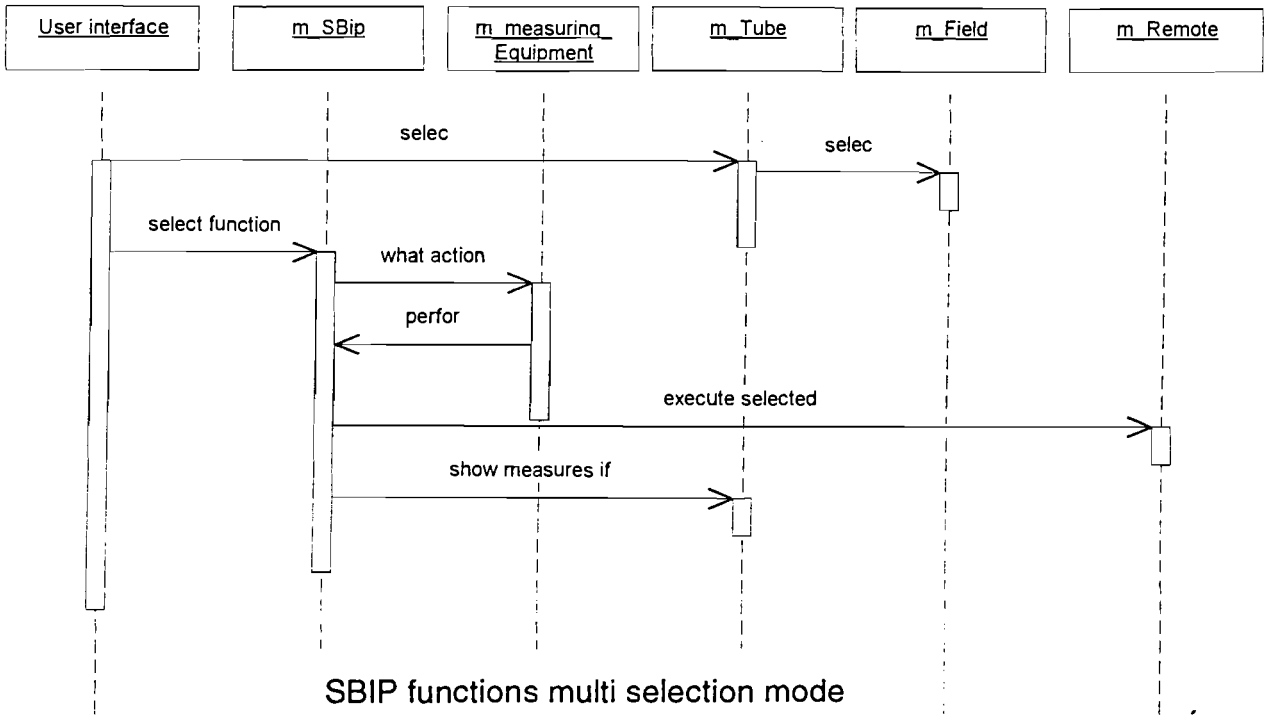
Measurement utils multi selection mode



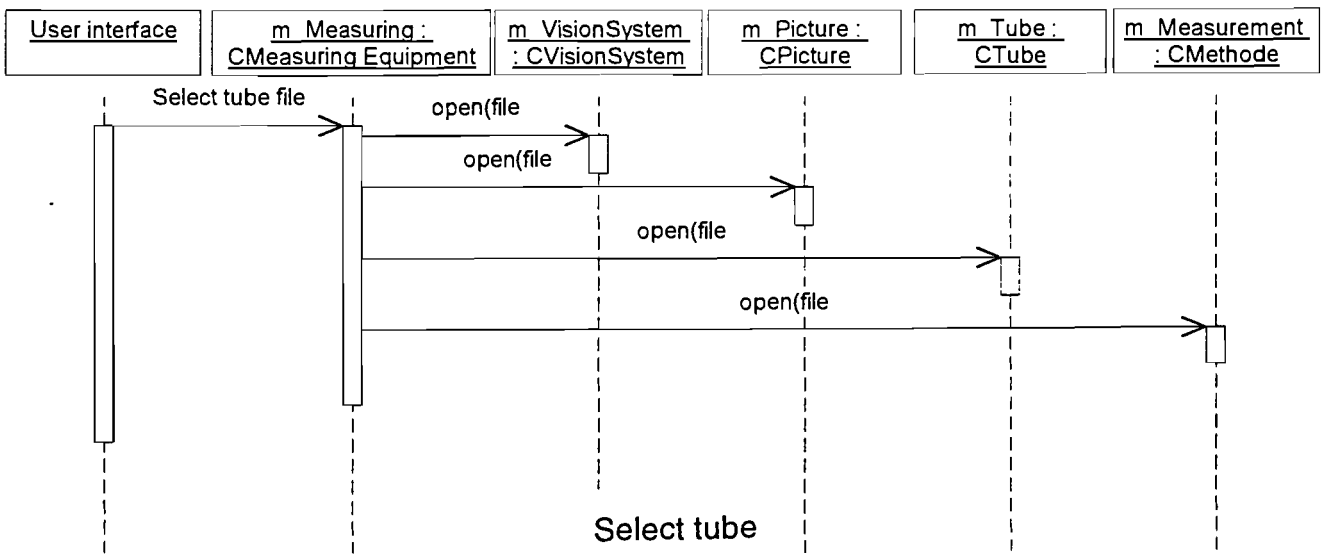
SBIP functions single selection mode



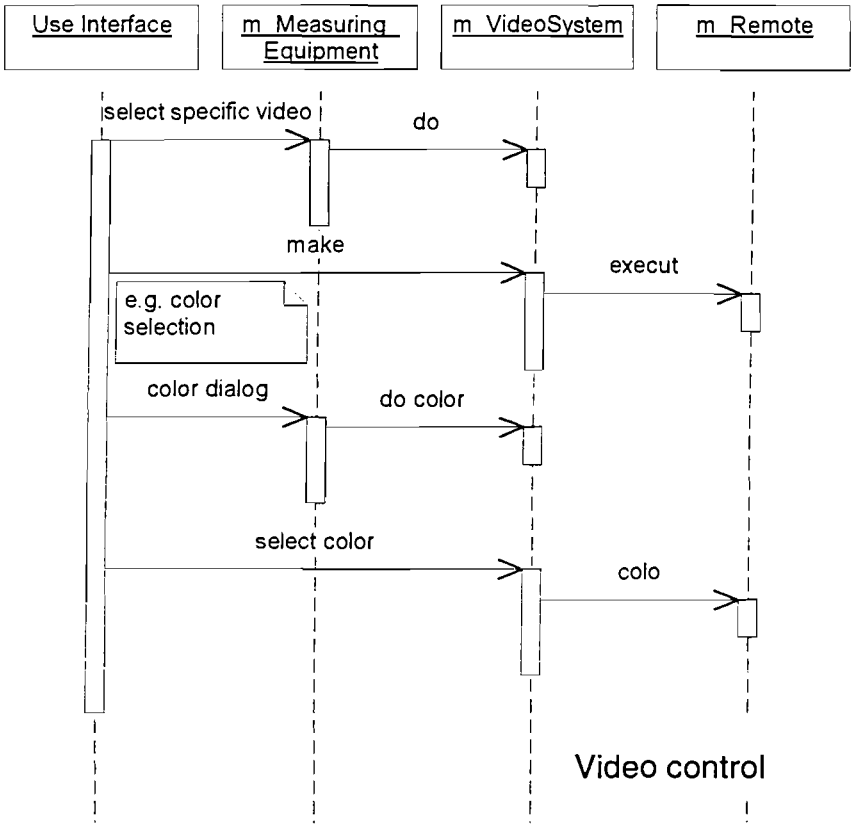
Measurement utils single selection mode



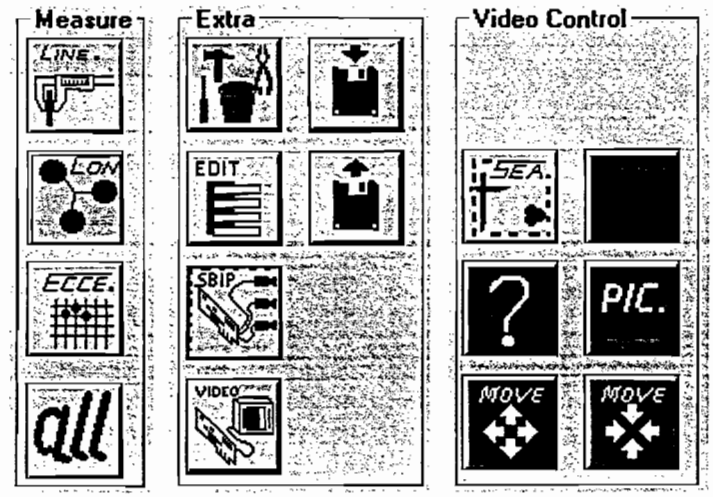
SBIP functions multi selection mode



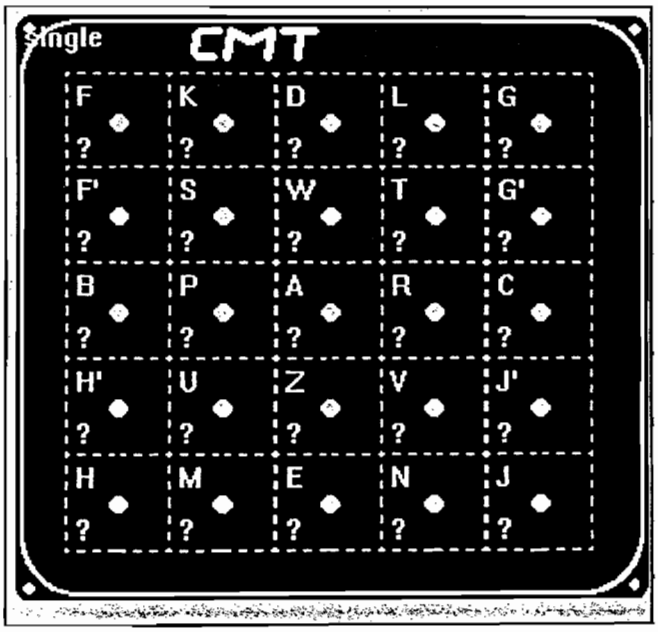
Select tube


Use Interface
m_Measuring Equipment
m_VideoSystem
m_Remote
select specific video
do
make
execut
e.g. color selection
color dialog
do color
select color
colo
Video control

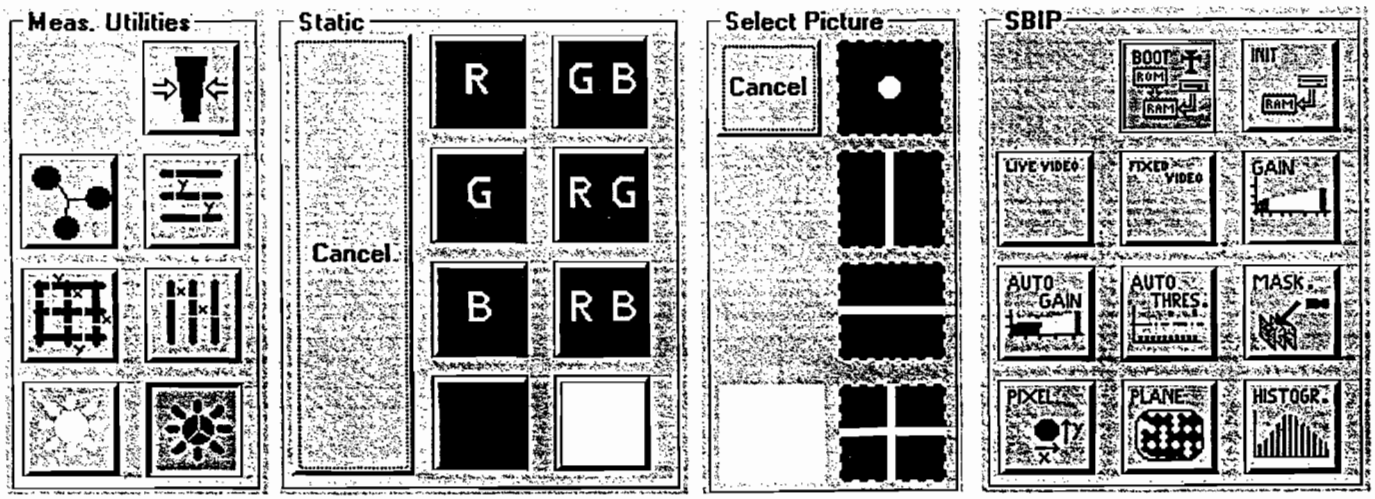
P. Dialogs



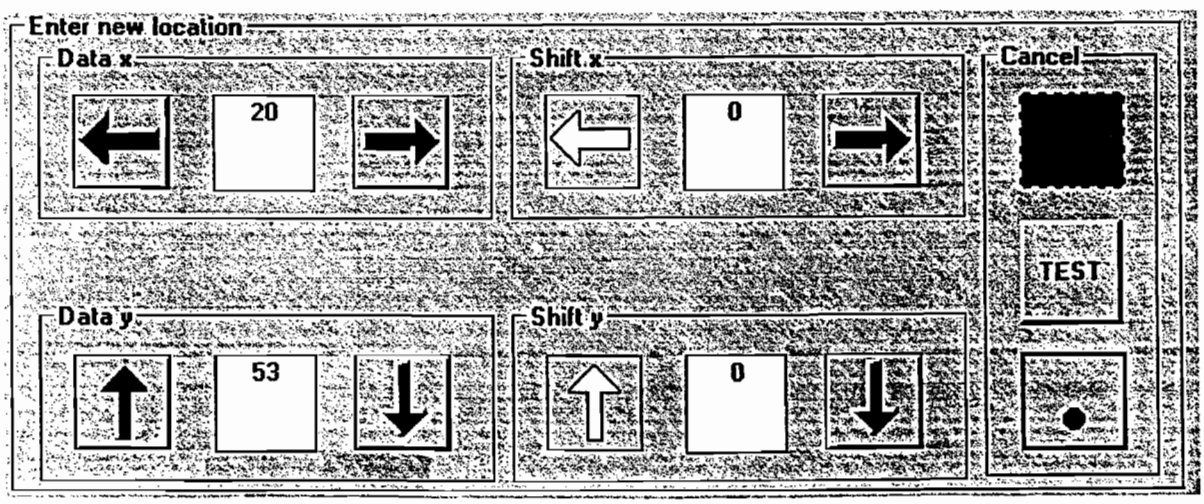
Measurement Extra Video



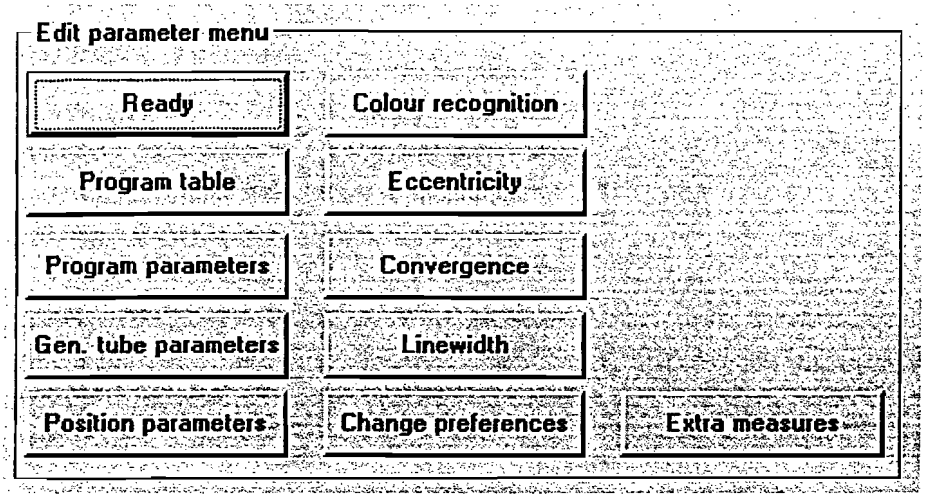
Field



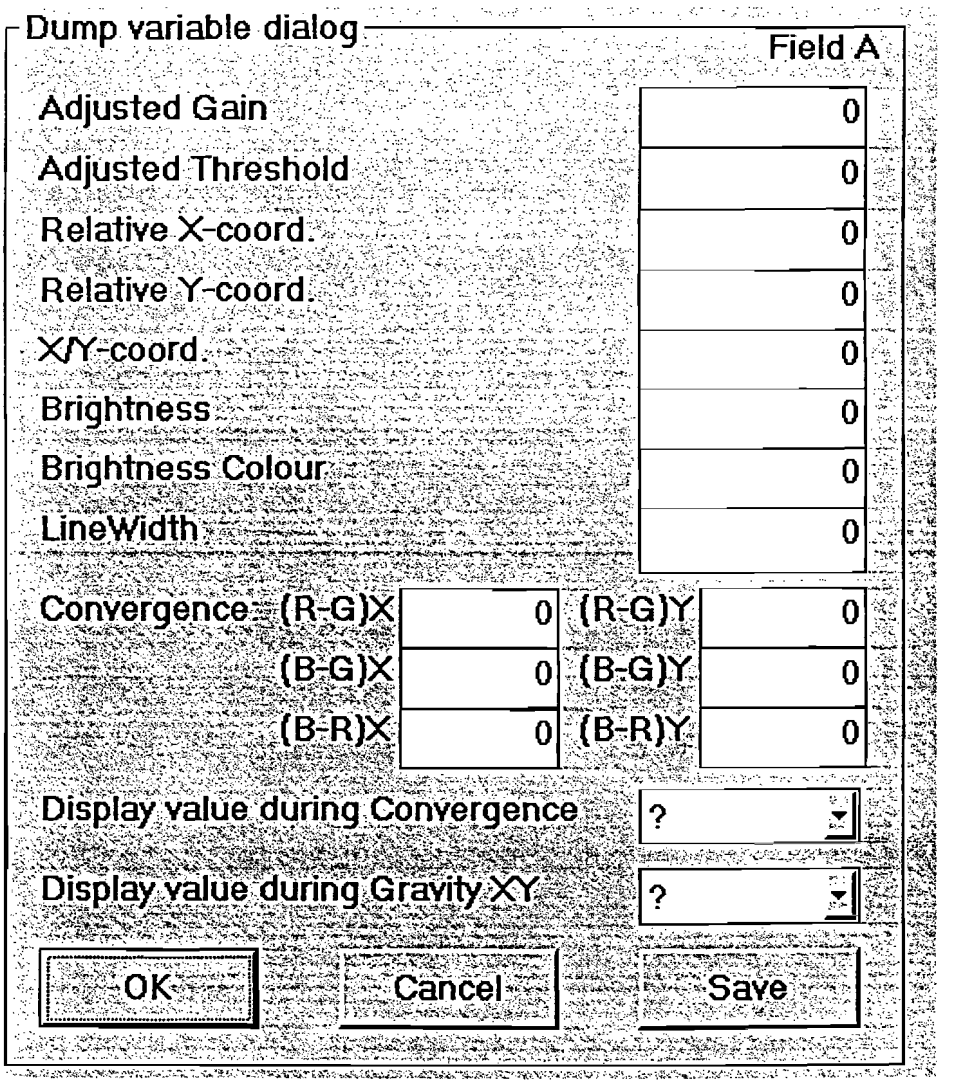
Measurement Utilities Colour Picture SBIP



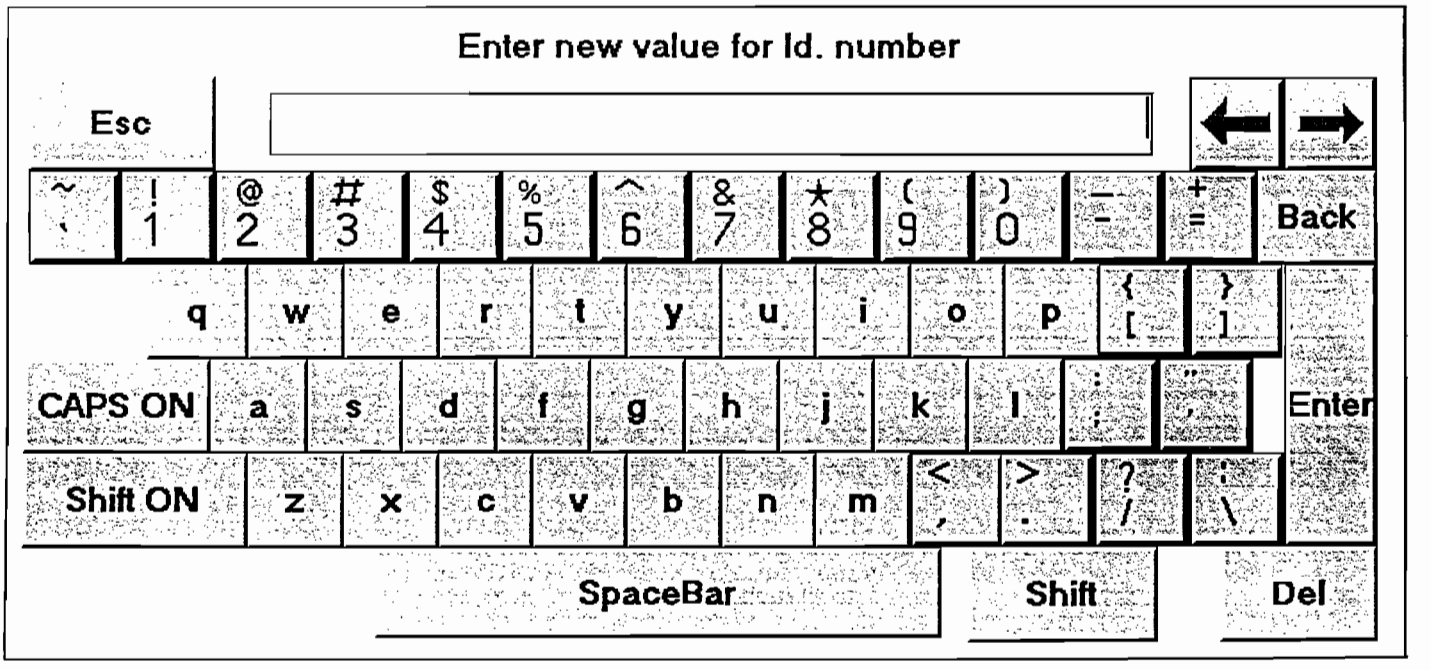
Move

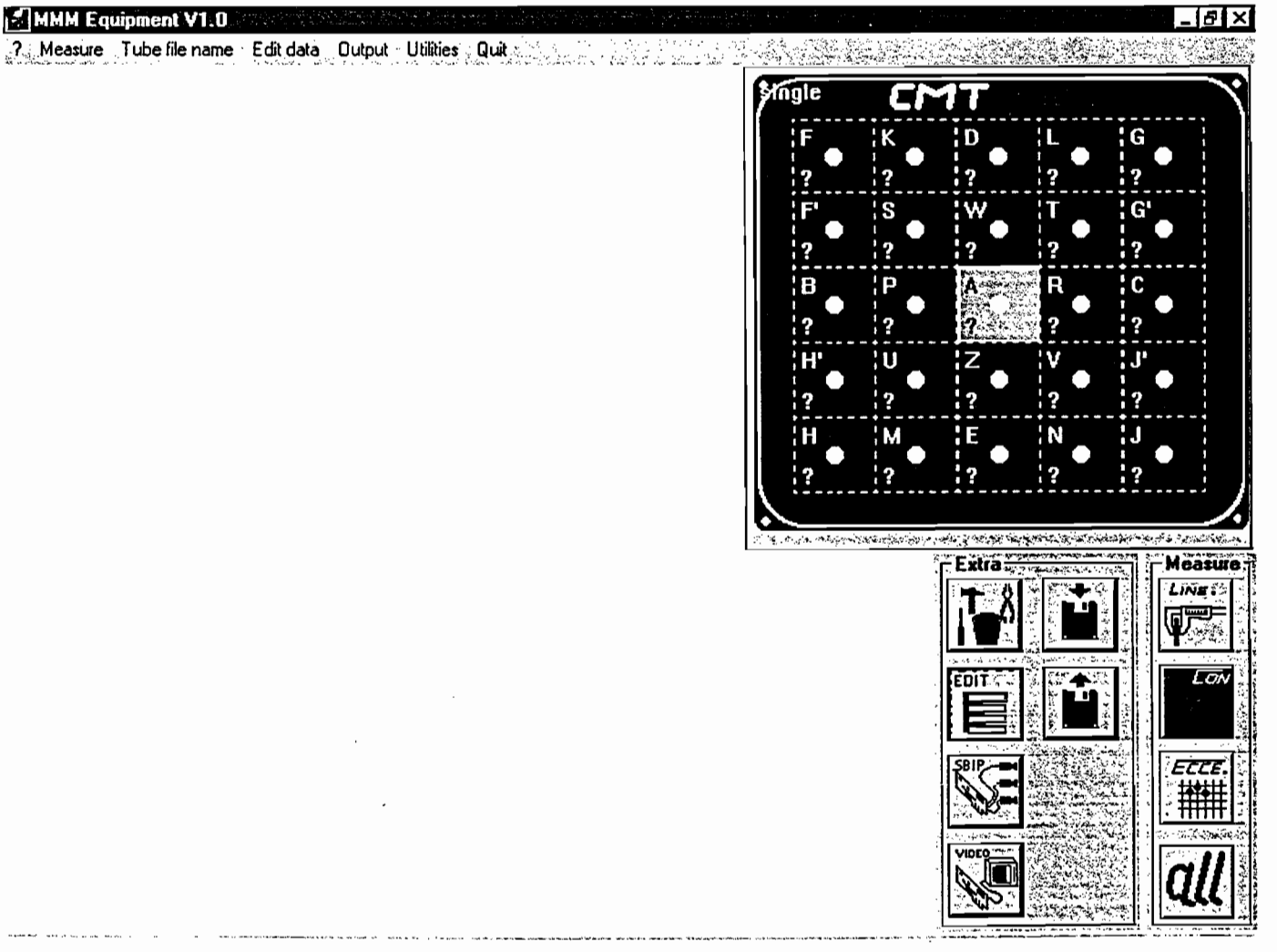


Edit Parameter Dialog



Edit Dialog


Enter new value for Id. number
Esc
Back
CAPS ON
Enter
Shift ON
SpaceBar
Shift
Del


MMM Equipment V1.0
? Measure Tube file name Edit data Output Utilities Quit
Single
CMT
Extra
Measure
EDIT
SBIP
VIDEO
LINE
LON
ECCE
all

Q. Example of part from parameter file

```
[LineWidthMethode]

DAFgun = No
Direction = XY
LevelX = 0
LevelY = 0
MicrostepStepSizeY = 0
MicroStepNrOfStep = 0
SensBlue = 0.000000
SensGreen = 0.000000
SensRed = 0.000000
Vg3dynNbtOfStep = 0
Vg3dynNom = 0
Vg3dynStep = 0
Vg3nbrOfStep = 0
Vg3Nom = 0
Vg3Step = 0

[TubeVariables]

Name =
Type = CMT
DeltaThreshold = 0
DeltaGain = 0
VideoSensFactorX = 0
VideoSensFactorY = 0
DeltaCentreX = 0
DeltaCentreY = 0
VideoShiftStepSizeX = 0
NrOfVideoShiftSteps = 0
PulseWidth = 0
1X2HorzFieldOfView = 0
1X2VertFieldOfView = 0
LineFrequency = 0
FrameFrequency = 0
ClockPulseType = Mains Locked

[FieldVariables]

FIELD = F
MaskPitch = 0

FIELD = K
MaskPitch = 0

FIELD = D
MaskPitch = 0

FIELD = L
MaskPitch = 0

FIELD = G
MaskPitch = 0

FIELD = F'
MaskPitch = 0

FIELD = S
MaskPitch = 0

FIELD = W
MaskPitch = 0

FIELD = T
MaskPitch = 0

FIELD = G'
MaskPitch = 0

FIELD = B
MaskPitch = 0

FIELD = P
MaskPitch = 0

FIELD = A
MaskPitch = 0

FIELD = R
MaskPitch = 0

FIELD = C
MaskPitch = 0

FIELD = H'
MaskPitch = 0

FIELD = U
MaskPitch = 0

FIELD = Z
MaskPitch = 0

FIELD = V
MaskPitch = 0

FIELD = J'
MaskPitch = 0

FIELD = H
MaskPitch = 0

FIELD = M
MaskPitch = 0

FIELD = E
MaskPitch = 0

FIELD = N
MaskPitch = 0

FIELD = J
MaskPitch = 0

[ColourRecognitionParameters]

FIELD = F
Gain = 0
Threshold = 0

FIELD = K
Gain = 0
Threshold = 0

FIELD = D
Gain = 0
Threshold = 0

FIELD = L
Gain = 0
Threshold = 0

FIELD = G
Gain = 0
Threshold = 0

FIELD = F'
Gain = 0
Threshold = 0
```